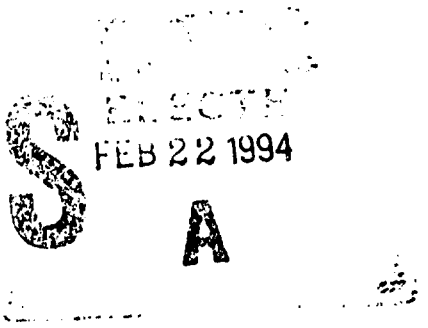
AFIT/GSS/LAS/93D-7

(1)

ADA275789



DEMONSTRATION OF IMPROVED SOFTWARE SUPPORT LABOR ESTIMATION FOR AIR FORCE OPERATIONAL FLIGHT PROGRAMS THROUGH FUNCTIONAL ORIENTATION

THESIS

Ronald L. Warner, Jr., Captain, USAF
Darrell L. Wright, Captain, USAF

AFIT/GSS/LAS/93D-7

Approved for public release; distribution unlimited.

The views expressed in this thesis are those of the authors and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

AFIT/GSS/LAS/93D-7

DEMONSTRATION OF IMPROVED SOFTWARE SUPPORT LABOR ESTIMATION FOR AIR FORCE OPERATIONAL FLIGHT PROGRAMS THROUGH FUNCTIONAL ORIENTATION

THESIS

Presented to the Faculty

of the School of Logistics and Acquisition Management

of the Air Force Institute of Technology

Air University

In Partial Fulfillment of the

Requirements for the Degree of

Master of Science in Systems Software Management

Ronald L. Warner, Jr., B.S., M.S.

Captain, USAF

Darrell L. Wright, B.S.

Captain, USAF

December 1993

Acknowledgements

We wish to thank our faculty advisors, Mr. Dan Ferens and Lt Col LaRita Decker, for their feedback and direction during the evolution of our thesis. Their support helped us discover our own misconceptions and reorient ourselves back to a productive path.

We also had a great deal of help in trying to obtain historical data for our thesis. The data we needed was not trivial to find and required some research. We are grateful to Paul Harbour, Nasser Ismail, James Peebles, and Jim Roberson for their time and energy in collecting the data we requested. We also thank all those who we called who weren't able to directly provide data, but who provided us with more contacts.

Finally, we wish to thank our wives, Pat and Faustina, for their understanding and support during our year-long effort to complete our thesis.

Ron Warner

Darrell Wright

Table of Contents

Page

List of Figures

List of Tables

Abstract

This study demonstrated two approaches to improve current software support effort estimation models for aircraft software. Both approaches involved a functional orientation not used by existing models. The first approach demonstrated how to orient a model to reflect the block change cycle modification process and how to represent support effort changes over time in order to improve effort estimation accuracy. Current software models do not reflect the support environment or the temporal characteristics of aircraft software support. The second approach demonstrated how to calibrate a model by properly selecting source data in order to increase accuracy. Support calibration is not addressed by current models. A literature search affirmed the validity of both approaches and the methodology. In addition, a standard description of the block change cycle was developed and validated. A prototype estimation model was derived from the COCOMO model and included a unique support calibration. Data was obtained from Air Force Software Support Centers but was unusable, so data was generated from the prototype for the demonstration. A method that was developed to compare the prototype with current models demonstrated that the prototype is an acceptable model.

DEMONSTRATION OF IMPROVED SOFTWARE SUPPORT LABOR ESTIMATION FOR AIR FORCE OPERATIONAL FLIGHT PROGRAMS THROUGH FUNCTIONAL ORIENTATION

I. Introduction

Chapter Overview

This thesis demonstrates methods of improving existing software support effort estimation models. This chapter begins with the importance of software support estimation in decision making. We discuss typical effort estimation methods in current software support estimation models and the expected accuracies of those models for Air Force software support projects using the block change process. Next, two hypotheses focus on objectives for improving the accuracy of software support estimation models. The scope of the research, definitions of important concepts, and a brief overview of the remaining chapters complete the Introduction.

General Issue

During any phase of a software project, decision makers must know the level of effort remaining for the project in order to make valid decisions (Lederer:51; Bourque:161). Before starting a software project, rational

decision makers normally weigh the expected benefits against the expected costs. The assessment of labor hour requirements is key to software project cost estimates. A good effort estimate must accurately aggregate the man-months required by specific skills into a total project requirement. An erroneous effort estimate leads to an erroneous cost estimate which, in a worst case, will lead to an erroneous start decision.

Once the project is selected, managers allocate resources to the project based on the cost estimate. This allocation directs how many people are hired. Without a good effort estimate, managers might hire extra people and waste money or hire too few people and fall behind schedule. During the project, managers must know how much effort has been expended versus how much effort remains in order to decide if their project is on schedule. Actual labor costs are accumulated and tracked against the estimated effort. An erroneous effort estimate can lead the manager to an unrealistic progress assessment.

As explained above, good effort estimation is essential to allow management to correctly select, staff, and monitor software projects. However, the pitfalls associated with poor effort estimation are not limited to the development of the software. Software support managers repeat the selection, staffing, and monitoring decisions made in development each time a change is made to the software. The impact of each individual decision is smaller than during development; however, the support phase of large software projects has an interesting characteristic. In large

software projects, 50 - 80% of the life-cycle cost occurs not during the development phase but during the support phase (after the software transfers to its operational site) (Banker:1, DSMC:7-1). So, in terms of cost, the support phase is at least as important as the development phase. Since the effort estimate may determine whether the project will start at all, managers should consider support effort very early when considering life cycle cost-benefit trade-offs. When predicting the cost-worthiness of a large software system, managers should also focus on the support of the software, not just the development.

In this thesis, we examine how to improve the relationship between computer software cost models and the software support environment. Although computer cost models are commonly understood, explaining our interpretation of software support can avoid confusion. Software support often is referred to as software maintenance, but this thesis uses the term software support because "support" avoids the connotation that maintenance is only corrective (Ferens1:3). Software support encompasses more than correcting coding errors; it also includes adapting existing code to interface with new hardware and perfecting code beyond its original capability (Hager:1638). Thus, the term software support better conveys the overall redevelopment aspect of adapting and perfecting software as well as correcting mistakes.

Another potentially confusing software support term is cost. Cost, a primary measurement of software support, is typically computed in man-

months instead of dollars. Man-months quantify the amount of effort required to complete a task while avoiding time-dependent conversions (such as inflation) encountered when measuring cost in dollars. Using man-months does not mean dollar costs are ignored since, given man-months, estimators can convert to dollars. Man-months are simply easier to compare among differing projects and times. In this thesis, the terms cost and effort are synonymous: both describe the number of man-months needed to accomplish a specified subset of the software life-cycle such as one block change during the support phase.

Typical Effort Prediction Methods

Software managers use many techniques to predict levels of effort. The techniques can be grouped into three general methods: asking an expert, comparing analogous projects, and invoking a software model (van Genuchten:32). The expert method simply tasks very experienced software analysts to predict the level of effort. Managers give experts the requirements of a system, and then the experts wield their experience and knowledge to forecast the cost of the new system. The analogy method estimates the cost of a project by comparing the cost of past projects to the cost of similar future projects. The implementor of this technique adjusts cost according to differences between the past and future projects. The last method, using a model, is the least used method (van Genuchten:37). This method uses mathematics to predict the level of effort for a given set

of input variables. The model is found by statistically comparing known project characteristics (inputs) with known project costs (outputs) (Banker:3). Assuming future costs come from the same statistical process as the past costs, estimators can find future effort levels from proposed project characteristics. For example, if given the number of lines of code and the complexity of a system, a software model, such as the basic COCOMO model, can output the man-months required to develop the project (Boehm:57). We will discuss the COCOMO model in Chapter II.

The prime advantages of model estimators are their objectivity (van Genuchten:40) and usability (Gulezian:235). A model is objective because it's a mathematical function, and a predefined function is resistant to bias. Nevertheless, the inputs to any function can be selected in a manner to bias the final result. While training and experience can help control this bias (Gulezian:42), models are also helpful because the required input and the computation process can be programmed into a software package. Non-experts can then use the software to provide quick effort predictions and adjustments.

Unfortunately, models also have limitations. The biggest disadvantage of models is a perceived lack of accuracy. Past predictions of software cost estimates from computer models indeed have been inaccurate. A thesis by a previous AFIT student (Ourada:1.3), conversations with current software program mangers, and our personal

experience confirm the poor performance of existing software cost estimation models.

Currently, the Air Force Cost Analysis Agency recommends four computer models for estimating software support efforts: 1) the Software Architecture Sizing and Estimating Tool (SASET), 2) the Software Evaluation and Estimation of Resources Software Estimating Model (SEER-SEM), 3) the PRICE-SL Software Estimating Model, and 4) the Revised Intermediate Constructive Cost Model (REVIC) (Mosemann:2). Each model employs statistical techniques to predict future software effort based on past efforts. The predictive capability of these models stems from the assumption that all the data from past projects come from very similar processes. However, the software engineering processes along with the historical data from those processes have constantly changed. Using historical data from changing processes yields computer models which predict with high amounts of variability. This high variability theoretically reduces the expected accuracy of a model's point estimation of effort. Any technique that can reduce the variability of a model's prediction can potentially increase the accuracy of the model's prediction.

Specific Problem

Before addressing how to increase model accuracy, the process by which computer models currently estimate effort should be addressed. All software support effort estimates from computer models lie on a continuum

between two opposing methods, parametric (top-down) and detailed (bottom-up) (Stewart:464). The heart of the parametric method is a model formed from correlating a measurable set of software characteristics to known levels of effort. Models based on the parametric method use the correlations to predict future efforts. Most current software support models tend to employ the parametric method.

The detailed method breaks down a large, complex process into small, simple steps and then sums an estimate of each simple step to predict the effort of the whole process. Each step should be small enough and simple enough to permit a quick, straightforward effort estimate of that step.

Estimation models need not use the parametric or detailed methods exclusively; models can combine both methods. A model could divide a process into a few large subprocesses, estimate each of these subprocesses by historical correlation (parametrically), and then sum the results. For example, REVIC, SASET, and SEER divide the life cycle of software into two parts, initial development and post deployment support. Each of these models uses a parametric model for development estimation and a slightly different parametric model for support estimation. The use of two models to estimate the software life cycle cost makes REVIC, SASET, and SEER more detailed than a single parametric model used to describe total software life cycle cost. However, the support phase estimates of REVIC, SASET, and SEER are purely parametric.

Armed with a basic understanding of the estimation process, we now discuss how to make the models more accurate. We propose that predictive accuracy of the software support models can be increased by functionally orienting the models to reduce the prediction variability introduced by the model. Functional orientation means adapting a model to better represent the function(s) of the software and/or the software support process. This functional orientation will be demonstrated in two ways, one for each type of estimation method.

The first way to functionally orient a model is to push towards the detailed end of the estimation continuum. The idea is to alter the support model to reflect the actual software support process. We altered the COCOMO model in a way which better reflects the actual process of software support used by the Air Force called the block change process. (The block change process is discussed in detail in the Definitions section later in this chapter.) Selecting the COCOMO model was the result of two considerations. First, REVIC, a model on the list of Air Force recommended models, was derived from COCOMO. Second, COCOMO was a well understood and popular non-proprietary model with good documentation. Therefore, we selected a baseline model familiar to Air Force estimators, thoroughly researched the baseline model, legally modified it, and then verified the modifications. Altering the COCOMO model encompassed consolidating the published regulations and common practices of the Air Force Materiel Command that described the block

change process. With a simple model of the block change process in hand, the next step was to adapt this process model to a COCOMO mathematical model to produce better predictions.

The second way to functionally orient a model is to keep the model on the parametric side of the estimation continuum but to limit the scope of the parametric correlation. The idea is to simply calibrate a model to a single functional area. The broadest functional area examined was aircraft operational flight programs (OFPs). We further divided OFP software into five functional categories: communication/identification, navigation sensors, core avionics, electronic combat, and offensive sensors. This calibration technique closely follows what many software experts advise - to calibrate the model to the local environment (Boehm:524; Thibodeau:6-6). Since the term local environment is poorly defined, this thesis interpreted local environment as a group of software programs that are functionally related. This interpretation avoided the connotation of environment as a local geographic location. Models calibrated to any of these subareas should predict better than models calibrated to aircraft OFPs or to models calibrated to all types of Air Force software. Both estimating approaches, functional calibration for parametric models and functional adaptation of more detailed models, reflect the basic engineering tenet that form follows function. We adapted this tenet to the software support estimation problem.

Hypotheses & Objectives

The research hypotheses and objectives are centered around the inflight software used aboard current Air Force aircraft. The aircraft are restricted to those with combat missions since combat aircraft have the greatest variety of software types.

Hypothesis 1. The functional, bottom-up construction of a software effort estimation model will increase the predictive accuracy of that model. The objectives are to

1) model the current block change functional process with simple, small steps.
2) program these steps into a computer model based upon COCOMO.
3) demonstrate the improved accuracy of bottom-up modeling.
4) find specific recommendations to tailor existing models for functional adaptation.

Hypothesis 2[1]. The functional, top-down calibration of a statistically-based software effort estimation model will improve the predictive accuracy of that model. The objectives are to

1) demonstrate the improved accuracy of functionally calibrating cost models.
2) find specific tailoring recommendations to improve the accuracy of cost models.

The acceptance or rejection of these hypotheses will not be conclusive, but they should help point the proper direction for improved software support cost estimation.

---

[1] Data limitations hindered assessment of this hypothesis. However, each objective is addressed to the extent possible.

Scope/Limitation of Research

Our research focused on estimating the main component of software support costs, man-months of effort, and not on auxiliary support costs such as facilities or hardware. These auxiliary costs are better addressed outside effort estimation models. Also, estimates for man-months of effort on such tasks as technical order changes and flight tests may vary between block changes and accuracy may vary accordingly. However, the proposed changes should increase accuracy over current models.

Evaluating the hypotheses required obtaining data from a sequential series of block changes on two different operational flight programs from each functional category. A time series of actual data could then be used to simulate an entire software support lifecycle to test the predictive accuracy of all the models for the entire software support phase with multiple block changes. Both time series and individual block change effort data could be used to calibrate the models and used to test the predictive accuracy of all the models for a single block change effort.

The minimum set of data sought for each block change was the number of lines of code changed, the time length of the block change, the effort duration in man-months, and a description of the code function. Other information, such as available memory and throughput, could fulfill a particular model's variable set. Most of the models require these additional variable values to fine tune the estimate; otherwise, default values must be used. Once entered, these additional variables were to be

held constant, except in the case of available memory and throughput. Unfortunately, acceptable size data was not available. The impact of this data unavailability is discussed in Chapters III and IV.

The data search was limited to combat aircraft (F-111, F-15, F-16, B-1, B-52, E-3, AC-130) that had an established software support history. These aircraft also had subsystems from many of the functional categories. Our search narrowed to these specific aircraft to focus on aircraft that were likely to have the data we needed. Although we expected few complete data sets to exist, we did expected to find, at a minimum, enough data to determine if we were pointed in the proper direction.

Narrowing the data search to specific aircraft was consistent with the focus of this thesis. After identifying a set of Air Force software that was functionally different than other types of Air Force software, we further divided that set, again based on function, into subsets. This thesis is a first cut at improving software support estimation. Future research still needs to be done once actual data becomes available to confirm the findings and to determine at what level of subdivision the proposed methods cease to work. This work provides few resolute answers but, instead, serves as a catalyst for future research.

Definitions

Functional Categories.

1) Core Avionics (Fire Control Computer, Stores Management, Display Generation) that are usually purchased from the airframe contractor as Contractor Furnished Equipment.

2) Offensive Sensors (Fire-Control Radar, Infrared Search Track sets, LANTIRN Targeting Pod, PAVE PENNY Pod, PAVE TAC Pod, AWACs radar, JOINT STARS radar) that may be internal or external to the airframe.

3) Communication/Identification Systems (HAVE QUICK, HAVE SYNCH, VHF, UHF, IFF, Integrated Air Data Modem, etc.) that are common across many aircraft types.

4) Electronic Combat Systems (Radar Warning Receivers, Chaff/Flare Dispensers, Electronic Jamming Systems, Missile Warning Systems, etc.) that are added to the aircraft for self protection.

5) Navigation Sensors (Global Positioning System, Inertial Navigation units, LANTIRN Navigation Pods, Terrain Following Radar, Microwave Landing System) that are used for precision navigation.

Block Change Paradigm. The software support lifecycle consists of a periodic series of redevelopment blocks as governed by DoD-STD-2167A and described by MIL-HDBK-347. These blocks may overlap or be separated in time and have the following characteristics:

1) Each block encompasses a mixture of support categories (maintenance, optimization, adaptation, new capability).

2) Each block may have a fixed interval, duration, budget, or manning level.

3) Each block is either organic (Air Force) or contractor logistic support (CLS) or a mix of both.

4) Each block requires access to non-aircraft facilities for development and test.

5) Facility costs (development and support) are part of the life-cycle cost and are not part of any particular block change cost.

6) Facility needs are similar within functions and different between functions.

7) Flight test costs for non-core avionics are budgeted by the aircraft and are not part of the change cost.

8) Change efficiency may be different for each block change.

9) Total effort per block change is a function of

a) number of lines to be changed
b) number of lines to be added
c) number of lines to be deleted
d) change complexity
e) available memory and throughput
f) organization efficiency
g) organization experience and skill level
h) organization resources
i) schedule
j) documentation

The actual process used when a new block change is produced is very similar to the development of new software. The biggest difference between a new development and a single cycle of the block change stems from the starting point and the requirements maturity. A block change starting point is an established software baseline. The starting point for a new development is a requirements document. The requirements for the block change are problem reports and change requests based on the actual performance of the software configuration and inherently different than the more abstract requirements that document new development uses.

These differences between a new development and a single block change result in an additional process at the start of each block change. In this process the block change requirements (problem reports and change requests) are reviewed, prioritized, and approved by the Configuration Control Board (CCB) for incorporation into the software baseline. From that point on, the support process is similar to a development effort. The requirements are mapped into the baseline software, coded, tested, and then released. After release, the process may begin again.

Common Terms.

1) Operational Flight Program (OFP): Software written for an airborne computer requiring real-time processing, interaction with other aircraft computers, and fault tolerance.

2) Lines of Code (LOC): Each OFP source instruction changed, added, or deleted in the software update. This should be counted in a before and after comparison.

3) Software Support Activity (SSA): the DoD or military service organization responsible for the software support of designated computer software.

4) Software Support Product (SSP): A single, fully developed, tested, documented, and supportable computer instruction set replicated in sufficient quantities and delivered to the installation point (excluding the original development). Under the block change process, SSP is usually completed on a periodic basis.

5) Block Change Cost Estimate: Estimation of all costs associated with the production and delivery of a single software support product.

6) Software Support Life Cycle Cost Estimate: Estimation of all costs associated with the production of all software support products for a specific computer.

7) Software Support Estimate: Estimation of all costs associated with the production of a single support product.

8) Bottom-Up Cost Estimate (detailed): Estimation method where the project is broken into a number of smaller tasks. Each task is estimated independently by analogy, by parametric model, or by best engineering judgement, and the results are summed for a project estimate.

9) Top-Down Cost Estimate (parametric): Estimation method where the project is estimated without significant subdivision of the tasks. The common methods of Top-Down estimation include analogy and parametric models.

10) Configuration Control Board (CCB): An organization composed of representatives from the SSA and software users that approves changes to the configuration of designated computer software.

Research Overview

Chapter II presents a review of available literature, regulations, and operating instructions pertaining to the Block Change Process and current methods used by Air Force organizations to estimate support costs. The next chapter also reviews available literature and model manuals to examine the paradigm that existing software cost estimation models use to estimate support costs and to examine alternatives to existing cost models. Interviews supplement the literature as needed. Chapter III addresses the methodology used to build and evaluate the models and the collection and analysis of the data. Chapter IV is devoted to findings while Chapter V contains our conclusions and recommendations.

II. Literature Review

Chapter Overview

The track record of cost models to accurately predict software support cost is disappointing. In a 1980 study sponsored by the Air Force, the Hughes Aircraft Company found that none of the current cost models fulfilled the requirement for estimating avionics embedded software support costs (Waina1:27). In 1991, Ferens concluded that none of the eight cost models he examined were shown to be quantitatively valid (Ferens1:11). Clearly, there is room for improvement in software support cost estimating models.

Before improvement can be made, a comprehensive understanding of the software support cost estimation problem is needed. This literature review investigates four areas: software support definition and process, general software cost estimation theory, current support model paradigms, and the COCOMO/REVIC software cost model. Each of these areas form a cornerstone to building better software support estimation models. The results of the literature review are presented in this chapter.

Software Support

Definition. The most relevant definition of software support comes from the Defense Systems Management College Mission Critical Resources Management Guide. The term for software support used in the Guide is

Post Deployment Software Support (PDSS) which has been defined by the Joint Logistics Chiefs (JLC).

> Post Deployment Software Support is the sum of all activities required to ensure that, during the production/deployment phase of a mission critical computer system's life, the implemented and fielded software/system continues to support its original operational mission and subsequent mission modifications and production improvement efforts. (DSMC:7-5)

Military Handbook 347, Mission Critical Computer Resources Software Support, defines PDSS as

> Those software support activities that occur during the full-rate production and initial deployment and operations support phases of the acquisition process. (DOD2:8)

Both definitions are compatible with the software support definition proposed in Chapter I. Software support is corrective, adaptive, or perfective. The JLC definition makes additional allowance for changes due to mission modifications. These changes are adaptive under the definition scheme used in this research.

Barry Boehm espouses the corrective, adaptive, and perfective categories to define support (Boehm:536). On the other hand, John Reutter divides support into seven categories: emergency repairs, corrective coding, upgrades, changes in conditions, growth, enhancements, and support (DSMC:7-6). The seven categories may provide a more descriptive categorization of the work, and further investigation may prove one categorization better than the other, but such investigation becomes an academic argument. The JLC definition succinctly shows the

overriding issue: software support is the effort required to make the software system continue to work after it is fielded in spite of mission and/or hardware changes. How software support is categorized, except for occurring before or after fielding, is generally irrelevant to the task of estimating the cost of the effort. Once the software has been fielded, any changes that follow are support changes no matter what other categorization is used.

Process. The usual process for making DOD software support changes is referred to as the "block change". Under this concept, a number of changes are made during one time span and then they are all released simultaneously (Ferens3:65). In 1980, the Avionics Laboratory of the Wright Aeronautical Laboratories sponsored the Hughes Aircraft Company to investigate the block change process for aircraft Operational Flight Programs (OFPs) as part of the Predictive Software Cost Model development (Waina 2). Hughes documented the support process for the A-7, F-111, F-16, F-15, various Electronic Combat (EC) equipment, and various pieces of Automated Test Equipment (ATE). The process described showed minor differences in detail from aircraft to aircraft and are typified by the F-16 process.

The F-16 block change process, according to the Hughes report, begins by collecting reports of computer program deficiencies and descriptions of new capabilities. The support organization then prepares preliminary Engineering Change Proposals (ECPs) for each potential

change. Next, feasibility studies and engineering tests are conducted to better define the change, and the results are presented at a Technical Conference (TC). Members of the TC establish priorities, revise the Preliminary ECPs as necessary, and obtain user approval. In the subsequent step, software requirements are formulated and then reviewed in a Preliminary Design Review (PDR). Programming and checkout occur in the next step culminating in a Critical Design Review (CDR). After CDR, the revised OFP is incrementally tested ending with flight test. The support organization performs Functional and Physical Configuration Audits at the end of the test phase. The software then goes through a Validation and Verification (V & V) process ending with the release of the software to the field (Waina2:344-345). It is interesting to note that the block change process documented in the Hughes report has changed little during the past 10 years.

Several changes in Air Force software regulations have occurred since the Hughes report was written, most notably the introduction of DoD-STD-2167 in 1985 followed by DoD-STD-2167A and -2168 in 1988. All of these standards affect the software documentation, configuration management, and quality programs of the software developer and the support organization. DoD-STD-2167A also requires the CDR before coding and adds a Test Readiness Review before testing (DOD1:10).

Other process changes are documented in MIL-HDBK-347. This handbook initiates the PDSS process with the submittal of a

problem/change request and then sequences four major phases: initial analysis, software development, system integration and testing, and lastly product logistics (DOD2:25). Spanning these phases is the continuous activity of support operations and maintenance. The final product of the process is a delivery package.

A more detailed description of the PDSS process would label the problem/change report as the name applied to any form that reports software problems or proposes software enhancements. For each report during the initial analysis phases, the software support activity (SSA) collects all necessary decision-making information including change classification, impact analysis, estimated effort, and risk identification. The Configuration Control Board (CCB) then examines this initial analysis and decides if the proposed changes should be implemented. The development phase accepts the initial analysis as a starting requirement and proceeds to develop the change until the modified software is ready for testing. During system integration and test, the SSA incrementally tests the system until the software performs acceptably on real-time hardware in a realistic operational environment. Faults are identified, isolated, and then corrected throughout testing. Finally, the SSA reproduces and verifies the final delivery packages, delivers them to the users, trains the users on the new changes, and may even install and check out the updated software. Throughout the entire PDSS process, support operations and

maintenance activities provide the overhead structure necessary to keep the SSA functioning smoothly.

It is interesting to note that the processes described by Hughes, DoD-STD-2167A, and MIL-HDBK-347 differ in only minor degrees. Although the PDSS process may require a few unique activities such as final package delivery and training, all the above processes are more similar than dissimilar. The support process descriptions demonstrate that the phrase "re-development" is an apt label for the PDSS cornerstone of the software support estimation. In fact, MIL-HDBK-347 explicitly cites the PDSS software development phase as following a DoD-STD-2167A development cycle (DOD2:25).

General Software Cost Estimation Theory

To aid understanding of the second cornerstone of software support estimation, general software cost estimation theory, an optimal software support estimation tool based on the available literature was synthesized. The first step was to define the characteristics of an optimal tool and identify what parameters were necessary for the tool to estimate effort accurately. The next step was to investigate the parameters themselves to understand their expected behavior over time and to understand their interactions with each other. The following section reviews the results.

Optimal Tool. An ideal software estimation tool would accept some number of known block change efforts along with their production

characteristics and would contain a regression relationship that could be used to predict future efforts. The production characteristics would be easily observable and would have been recorded along with the original data. The most likely factors to be correlated are magnitude of effort (expressed in man-months) and the size of the product (expressed in LOC). Given a set of expected lines of code and production characteristics, the tool could find an estimate of the effort required. Any variations in man-hours between projects of the same size are due in part to the differences in the situation under which the effort is accomplished (its production characteristics). A better tool reduces these variations by adjusting for the production characteristics. If possible, the tool should divide the variation sources into mutually exclusive categories. Otherwise, if covariance exists between variation sources, the tool must perform additional calculations.

The number of potential production characteristics is enormous so some reasonable limit must be found. What is the minimum set of data needed to build a model? Given the definition and process of software support, any software support cost estimation tool needs to account for at least the following parameters:

1) the magnitude of the software change (size).
2) the ease of altering the software (maintainability).
3) the organizational efficiency of changing software (productivity).
4) the time allowed for the change (schedule).

These parameters are not independent. Some dependencies are intuitive. For example, schedule may impact size (management may

reduce the scope) if the estimate shows the work can't be done within the schedule. Productivity should increase for easily maintainable software. Other interdependencies emerged from the literature as well. Furthermore, the first three parameters can be divided into a number of factors that should be addressed separately. These factors as well as the parameter interdependencies are discussed below.

Size. Since size is the primary cost driver for software projects (Boehm:58), size is often the starting point of a software cost model. Mukhopadhyay and his associates assert that "A fundamental problem of software estimation is the determination of software size" (Mukhopadhyay:156). While Boehm refers to "Annual Change Traffic" as being equivalent to development product size, he says little about how to derive it (Boehm:536). For software support changes, the change needs to be expressed in some term of size that accounts for the number of lines of software to be changed. Software size is normally expressed as Lines Of Code (LOC), Source Lines Of Code (SLOC), or Deliverable Source Instructions (DSI). All three units basically refer to a single line of code as might be seen on a code printout.

After a line size unit is identified, a choice of which lines to count or not to count is needed to further define size. Low and Jeffery list five counting variations: 1) count only new lines; 2) count new lines and changed lines; 3) count new lines, changed lines, and reused lines; 4) count

all delivered lines plus temporary scaffold code; and 5) count all delivered lines, temporary code, and support code[1] (Low:64).

Of these five variations, variation 2 is preferred since it best captures only what was changed from the baseline to the final block change product with the exception that this variation does not account for deleted lines. Variation 1 gives a count of zero if no new lines were added. Variation 3 captures one complexity facet but counts more than what was changed. Counting reused lines can capture the added complexity of designing and checking for potential problems in a large program. However, this same complexity facet can be captured by calibrating the model to similar sized programs. Variations 4 and 5 not only count more than what was changed, but they also require extra code counts to capture the size of temporary development code. Temporary development code is not part of the baseline and is not delivered. Once past the sizing units and line selection, an estimate of the size of the change is still needed.

According to current literature, there are three general methods of deriving size which are available to the estimator. The first method is analogy in which a similar effort of known size is selected and its size becomes the estimate (Reifer:159). The second method is expert analysis where an expert estimates the size of the effort based on previous experience (Reifer:159,160). The third method, Function Point analysis, is

[1] Support code used in this context is code which supports development (such as stubs, drivers, or tests) and not code which is produced during the support phase.

a process that uses software functions to predict size and complexity (Reifer:159). All three of these methods typically have been applied to estimating development software size. However, software support sizing can have unique differences that might not exist during development size estimating.

Once a software product enters support, three changes can occur to the baseline code. Lines of code can be added, modified, or deleted. Size measurements historically have focused on the original development size by creating new code where none previously existed. However, software support does not always develop completely new requirements and a code sizing technique for support is needed for code modification and deletion activities. Two techniques that consider size for changes other than new code were found in the literature.

The first technique addresses code modification and comes from Boehm's conversion cost estimating relationship (Boehm:558). For this research, converting old code to a new application is tantamount to modifying code within an existing application. The equivalent delivered source instructions (*EDSI*) for a number of adapted DSI (*ADSI*) is found by multiplying the latter by a conversion adjustment factor (*CAF*) (see equation 2.1).

$$EDSI=(ADSI)\frac{CAF}{100} \qquad \text{Eq. 2.1}$$

where *EDSI* = equivalent delivered source instructions
ADSI = adapted delivered source instructions
CAF = conversion adjustment factor

The *CAF* can be viewed as a percentage fraction of the adapted (modified) code size and is the sum of two parts, the adaptation adjustment factor (*AAF*) and the conversion planning increment (*CPI*). The AAF is found by calculating a weighted average of the percentages of design modified (*DM*), code modified (*CM*), and integration required for modified software (*IM*) (see equation 2.2).

$$AAF=0.40(DM)+0.30(CM)+0.30(IM) \qquad \text{Eq. 2.2}$$

where *AAF* = adaptation adjustment factor
DM = % design modified
CM = % code modified
IM = % integration for modified software

Boehm selected weightings based upon a general average fraction of effort devoted to design, code, and integration/test (Boehm:137). The *CPI* value is found by using a simple table developed by Boehm. The table is shown in Table 2.1.

Table 2.1

CPI Values (Boehm:558)

CPI Value	Level of Conversion Analysis and Planning
0	None
1	Simple conversion schedule, acceptance plan
2	Detailed conversion schedule, test and acceptance plans
3	Add basic analysis of existing inventory of code and data
4	Add detailed inventory, basic documentation of existing system
5	Add detailed inventory, detailed documentation of existing system

As mentioned earlier, $CAF = AAF + CPI$. Using this relationship and substituting equation 2.2 for AAF, a more detailed form of the $EDSI$ equation is derived (equation 2.3) as shown below.

$$EDSI = (ADSI)\frac{(0.40*DM + 0.30*CM + 0.30*IM) + CPI}{100} \qquad \text{Eq. 2.3}$$

where $EDSI$ = equivalent delivered source instructions
$ADSI$ = adapted delivered source instructions
DM = % design modified
CM = % code modified
IM = % integration for modified software
CPI = conversion planning increment

Once computed, an $EDSI$ can be substituted in cost model equations in place of a pure development DSI measure.

The second technique that sizes code other than new code comes from Reifer Consultants in their manual for SoftCost-R (Kane:R-83). This sizing model differentiates among five categories for a support line of code.

A line can be new, added, deleted, modified, or retested. The model also distinguishes between actions that occur upon a single LOC and those within a module of code. Besides differentiating between lines and modules, there are two subtle differences between Low and Jeffery's change categories (page 2.9) and those proposed by SoftCost-R. The first is a category to count code that is not changed but is retested during the block change process. The second difference is a splitting of the added category based on the source of the added line. The SoftCost-R new category represents a line created from scratch for the block change while the added category represents a line created for some other program that is added (reused) as part of the block change. The equivalent size is the sum of the size of new code plus the size of reused code (see equation 2.4).

$$Equivalent_{Size} = New_{Size} + Reused_{Size} \qquad \text{Eq. 2.4}$$

Each size category is weighted as shown in Table 2.2.

Table 2.2

Reused Code Weightings (Kane:R-83)

Percent Weight	Type of Reused Code
27	Modified Modules
15	Deleted Lines
53	Added Lines
24	Changed Lines
11	Deleted Modules
17	Retested Modules

Reifer includes a set of assumptions for e· ·n of the five categories to allow estimators to properly categorize changed lines (Kane:R-84). The assumptions are listed below:

- New code will be developed according to a well-defined process and set of product standards, including those for documentation.
- Reused code may not necessarily have been developed according to the well-defined process and products standards. Its documentation may or may not be up-to-date.
- Reused code will be identified during the preliminary design phase so that code added, deleted, and/or changed within units will go through all subsequent life cycle activities.
- Deleted lines require reduced design, coding, and documentation effort, and no testing of the deleted lines.
- Changed code will take the same implementation effort (i.e., detailed design, coding, and unit testing) as new code in proportion, but with less documentation and testing.
- Reused code will not include code added, deleted, or modified in any other way.
- Retested, unmodified code will require revalidation of the interface design and retesting activities only.
- All modified and reused code will be completely retested and requalified prior to integration into the system.

The total of the reused code is the weighted sum of all the appropriate code types as shown in Table 2.2. The following equation is the complete expansion of equation 2.4 according to Table 2.2.

$$Size_{Equivalent} = New_{Lines} + [(.27 Modified_{Modules}) + (.15 Deleted_{Lines}) + (.53 Added_{Lines}) + (.24 Changed_{Lines}) + (.11 Deleted_{Modules}) + (.17 Retesed_{Modules})] \quad \text{Eq. 2.5}$$

Each part of the equation should be expressed in the same units. Generally this will be LOC, although it could be a percentage of the total code. Equation 2.5 shows that added LOCs are weighted less than new LOCs. In addition, since the weightings total more than 1.0, this equation

allows a highly modified portion of code to cost more than functionally equivalent code developed from scratch.

This literature review on size has shown that any size estimate involves selecting units, an appropriate set of LOCs, and an overall sizing methodology. For a support size estimate, Boehm's conversion relationship and Reifer's sizing model were presented as tools that can account for size other than new code. The major point to be made about size estimation is that better size estimates produce better cost estimates. Reifer states "Because most of the popular software cost estimating models in use today are extremely sensitive to size inputs, there is a direct correlation between improving the capability to predict both size and cost" (Reifer:159). Therefore, size estimation should provide the foundation of the estimate and the remaining parameters of maintainability, productivity, and schedule should be used to fine tune the estimate. The next subsections address these remaining three parameters.

Maintainability. Maintainability is a design feature of the software, its documentation, and its environment. The design sets the structure of the code under which support programmers can alter and maintain the code. The documentation communicates this structure and the software environment bounds the programmer's freedom to alter the code. Therefore, these three characteristics affect maintainability and help determine if the software itself has been designed to allow easy modification or if the software documentation understandable and current.

It is easy to underestimate the effect of documentation on support effort, but according to Robson and his coauthors, "50-90% of maintenance time is devoted to program comprehension" (Robson:79). The authors go on to state that comprehension can be affected by the design of the software, the style in which the software was written, the convention followed when naming variables in the software, the presence of indentation and the number of spaces used in the format of the software, and the presence of comments to explain the software (Robson:80,81).

Robson and his coauthors also discuss automated systems that have been developed to help the maintainer better understand the software but drew no conclusions on the usefulness of those systems (Robson:80-83). However, there is an interdependency between the maintainability parameter and the productivity parameter. Does the maintenance organization have and use such automated systems? If so, what is the effect on productivity? Answers to these questions need to be obtained to properly tune the effort predicted from size alone. The discussion so far has dealt with the software itself and has excluded the documentation of the software. The documentation presents other potential pitfalls which are discussed in the following paragraphs.

Documentation must not only be understandable it must be precise enough to prevent misunderstanding. Cioch states

> In practice, when one wishes to ascertain the understandability of a particular software-related product, one is often concerned not only with the degree to which, or the ease with which, the information is

> grasped mentally, but also the degree to which it is misinterpreted by the person examining the product. (Cioch:85)

He suggests that misinterpretations are more dangerous than lack of comprehension. Misinterpretation is harder to detect and can cause unintentional changes to the specification which result in software that doesn't match what the user expects. When misinterpretation mistakes go unnoticed in the current change, they must be fixed later (Cioch:86).

Another possible constraint on maintainability is the hardware environment where the software resides. In real-time environments such as avionics, software is especially susceptible to memory and throughput constraints imposed by hardware. As the operating memory available decreases, programs have less room to expand for corrections or enhancements. This lack of room forces support programmers to write more size-efficient code and increases the effort needed to write the code. Throughput responds similarly. As the available throughput of software decreases, programs have less time to manipulate data and communicate with other programs. This restricted ability also forces support programmers to write more efficient real-time code in terms of throughput (or timing). Again the effort required increases. In a continuation of the Hughes study of the Predictive Software Cost Model, SYSCON quantified these effects for several different support phases (see Table 2.3). In Table 2.3, the timing fill is equivalent to throughput.

Table 2.3

Constraints of Memory and Timing (SYSCON:33)

Support Phase	% Memory Fill	% Timing Fill
Requirements Review	$1.42X^{.808}$	$1.33X^{.624}$
Design	$2.00X^{1.50}$	$1.82X^{.130}$
Development	$1.88X^{1.35}$	$1.82X^{1.31}$
Integration	$1.59X^{.978}$	$1.55X^{.904}$
Test & Evaluation	$1.32X^{.607}$	$1.39X^{.688}$
Documentation	$1.13X^{.307}$	$1.09X^{.182}$
Reproduction/Installation	$1.04X^{.095}$	$1.04X^{.148}$

For both restrictions, the equations apply only if the percentage is greater than or equal to 75%. SYSCON determined that lower percentages had no effect on cost (SYSCON:41).

The PRICE-S model describes the effects of available speed (time) and memory utilization as one relationship (see Table 2.4). This model shows that the speed and memory constraints have a much greater effect on cost than on schedule. In Table 2.4, normalized costs greater than 1.00 represent an effort increase beyond that for an unconstrained effort. Utilization ratios less than 0.50 have no effect on cost.

Table 2.4

Utilization of Available Speed and Memory (Boehm:516)

Utilization	Normalized Cost	Normalized Schedule
0.50	1.00	1.00
0.60	1.08	1.00
0.70	1.21	1.00
0.80	1.47	1.05
0.85	1.73	1.10
0.90	2.25	1.18
0.95	3.78	1.35

Productivity. Following maintainability, the second major parameter needed to fine tune an effort estimate is productivity. Productivity is the measure of an organization's efficiency of converting a conceptual change into reality. The less time and resources required, the more efficient the organization. Normal organizational inefficiency takes the first bite out of productivity. Fried states

> People in formally organized groups cannot be productive 100% of the time for extended periods. According to general overhead estimates, in the average organization, at least 25% of employee time is required for vacations, sick leave, personal time off, training, coffee breaks, and administrative and organizational meetings. In addition, 10% of employee time (a conservative estimate) is nonproductive because of late completion of activities on which the employee depends, poor work scheduling, personal conversation, and other forms of idle time. (Fried:28)

There are a number of conceptual factors that affect productivity of an organization engaged in software maintenance. Obvious factors which affect Air Force SSAs include the experience level of both management and

maintainers, maturity of the maintenance process, software engineering practices and methods, familiarity with the software being modified, and resources available (computer time, specialized software tools, etc.). Banker, Datar, and Kemerer cite over 100 variables to explain productivity (Banker:1). However, their productivity model uses only five variables: the ability of the project team members, the level of previous experience with the application, the use of structured analysis and design software methodology, the level of hardware response time, and the operational quality of the resulting system (Banker:6). The last variable is not used to predict productivity but rather to determine the quality of the products produced. The obvious factors affecting Air Force SSAs listed earlier in this paragraph have only one element in addition to Banker's model. This conceptual agreement is very good considering Banker's model is a research tool used to measure the productivity of software maintainers working for a commercial bank. Even though the applications and language used for business are not what would be used in Air Force weapon systems, the models suggest that a small set of variables might be used to estimate productivity.

A benefit of successive block changes for software supporters is the built-in educational experience that occurs through those successive block changes. SYSCON quantified this relationship for different support phases (see Table 2.5) and reported that the maximum time of this effect is six years (SYSCON:44). At the end of six years, the supporters should

have learned the maximum amount from the original code. This relationship assumes that the software supporters are not the same programmers who developed the software.

Table 2.5

Learning Effect by Phase (SYSCON:33)

Support Phase	Years of Support
Requirements Review	$1.61X^{-.361}$
Design	$1.64X^{-.375}$
Development	$1.65X^{-.374}$
Integration	$1.65X^{-.374}$
Test & Evaluation	$1.58X^{-.330}$
Documentation	$1.43X^{-.258}$
Reproduction/Installation	$1.19X^{-.374}$

Symons confirms this initial productivity increase in his description of DuPont's support productivity versus system age (Symons:149). DuPont's productivity improved from about 1.5 work-hours per function point to about 0.25 work-hours per function point[1] in the span of about 3 years. However, after those initial three years, the productivity slowly decreased to near 1.0 work-hour per function point during the next 15 years. As plotted in Figure 2.1, these two trends exhibit a classic "bath-tub" shape, an initial decrease in required effort (increase in productivity) followed by a steady effort increase (decreasing productivity) for the

[1] A function point is another sizing method. In this case, one function point can be considered proportional to one line of code. (See Symons for more information on function points).

remaining support life span (Symons:150). Symons attributes the slow rise in required effort to technical deterioration because the original pristine design becomes degraded from constant maintenance and enhancement (Symons:150).

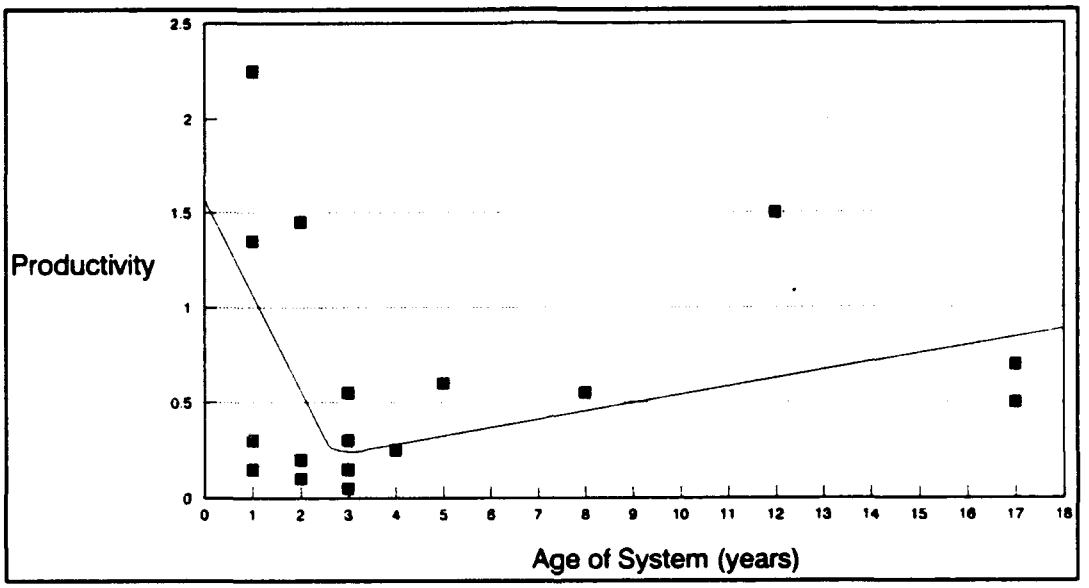


Figure 2.1 - Support Productivity versus System Age (Symons:149)

Lehner also describes this bath-tub phenomenon as an initial 3-year decrease in the corrective portion of maintenance from about 25% to 15% followed by a slower but steady 9-year increase up to 45% (Lehner:135). Lehner points out that this behavior is more prevalent within environments where the requirements are constantly changing, and less prevalent where the requirements are static (Lehner:137).

A third observer of design deterioration is Yuen. He confirms decreasing productivity with age in his evolution dynamics report. He

states that "the original structure of a piece of large software is inevitably corrupted each time the software is modified" (Yuen:160).

Schedule. The last major fine tuning parameter, behind maintainability and productivity, is schedule. This variable strongly interacts with productivity. Fried espouses that schedule is a balance of costs. Shortening or lengthening a schedule almost always increases the cost. Lengthening a schedule also delays the benefit of a new system (Fried:28). Fried argues that schedule should be driven by team size because each member in a programming group must spend some time communicating and interacting with others in the group (Fried:28). Thus, an excessively large group requires an excessive amount of communication. According to Fried, the formula for computing the number of possible interactions (I) between a group of K people is given in equation 2.6.

$$I=\frac{K(K-1)}{2} \qquad \text{Eq. 2.6}$$

where I = number of possible interactions
K = number of people in a group

Fried then uses the equation in an example of 90 people working a standard 40 hour work week. Out of 3,600 available hours, only 538 are productive. This is independent of how the 90 people are organized (Fried:28-29). Groups of 10 people or less are the most productive while larger groups spend more time communicating for each additional team member (Fried:29). Fried suggests that using proper tools and modern

programming techniques reduces the number and length of communications (Fried:35).

The support schedule, whether driven by team size or management dictate, combines with productivity, maintainability, and size as minimum parameters of an optimal estimation tool. Of these four parameters, size is the most important. The other three can be considered to fine tune an effort prediction based solely on size. How a specific model accounts for these parameters determines the unique characteristics of that model. The next section briefly examines how the Air Force recommended models handle these parameters.

Current Support Model Paradigms

As stated earlier, there are four models recommended by the Air Force Cost Analysis Agency. They are 1) Revised Intermediate Constructive Cost Model (REVIC), 2) the System Evaluation and Estimation of Resources (SEER), 3) the Software Architecture Sizing and Estimating Tool (SASET), and 4) PRICE-SL. The REVIC, SASET, and SEER models represent software support size as a constant number of LOC changes made to the software per year (Silver2:3-12; REVIC:1.4; SEER:6-5). The term used in this research for support size, Annual Change Traffic (ACT), is borrowed from COCOMO. The ACT concept is simple in that the estimator determines what percentage of total SLOC will be changed per year and provides this to the model. ACT models a

system under continuous change. The best interpretation for a system employing the block change process for support is a continuous series of block changes with each effort being identical in size and taking exactly one year to complete. None of the models is flexible enough to handle variation from the one year schedule.

As for the maintainability parameter, only SEER accounts for decreasing maintainability from memory and throughput. For productivity, none of the models seem to account for learning curve effects or for design entropy effects, nor do they capture the "bath-tub" curve supported by the literature. Flexibility to handle various schedules, changes in memory and throughput, a learning curve, and design entropy is necessary to model the real world. REVIC/COCOMO algorithms can be adapted to accept this flexibility.

COCOMO Model Description

Of the four Air Force recommended models, both SEER and PRICE-SL are proprietary models and are not subject to unconstrained alteration. Of the remaining two, SASET and REVIC, REVIC is much easier to calibrate (Ourada:4.8). Since a model which could be freely altered and easily calibrated was needed to test our hypotheses, model research was oriented towards understanding the algorithms and construction of REVIC. REVIC is essentially an implementation of Boehm's COCOMO with a few additions. Therefore, a discussion of

COCOMO is warranted since it directly contributes to an understanding of REVIC. The COCOMO model, as published in Barry Boehm's *Software Engineering Economics*, has three levels of detail: the basic, intermediate, and detailed levels. Each of these COCOMO levels are described in the following sections.

Basic Description. The basic COCOMO model predicts development man-months from one input. That input is size measured in thousands of delivered source instructions (*KDSI*) (Boehm:57). The equation for the basic model is shown in equation 2.7.

$$MM = a(KDSI)^b \qquad \text{Eq.2.7}$$

where MM = Man-Months
a = coefficient factor
b = exponent factor

This equation reflects the exponential characteristic of large systems. If a software system doubles in size, the effort needed to produce that system more than doubles. This non-linear effect helps estimators predict a variation in man-months given a variation in *KDSI*.

Since equation 2.7 is the basic COCOMO equation, it should only be used for first-cut approximations (Boehm:114). Although this equation can account for large-scale variations in the characteristics of any project, the parameter pair a and b can be changed to accommodate three basic modes of a project environment: the organic, semidetached, and embedded modes (Boehm:78-79). These basic modes are summarized in Table 2.6.

Table 2.6

Project Modes for the Basic COCOMO Model (Boehm:75-85)

Mode	Description	Equation
Organic	- Thorough understanding of project - Highly experienced personnel - Stable and familiar environment - Minimal need for innovation - Low premium on early completion - Relatively small size (less than 50 KDSI)	$MM=2.4(KDSI)^{1.05}$ Eq.2.8
Semidetached	- Considerable understanding of project - Intermediate or mixed personnel experience - Partial experience with project aspects - Larger sizes (less than 300 KDSI) - Mixture of Organic and Semidetached	$MM=3.0(KDSI)^{1.12}$ Eq. 2.9
Embedded	- General understanding of project - Tight constraints - Complex hardware & software - All sizes - Take up hardware slack	$MM=3.6(KDSI)^{1.20}$ Eq. 2.10

Intermediate Description. For second-cut approximations, the intermediate COCOMO equation (see equation 2.11) contains 15 more inputs to increase the accuracy of the effort estimate (Gulezian:237).

$$MM=a(KDSI)^b \prod C_{ij} \qquad \text{Eq. 2.11}$$

where MM = man-months
a = coefficient
$KDSI$ = thousands of delivered source instructions
b = exponent
$\prod$ = product function
C_{ij} = adjustment, i = attribute & j = selection

The C_{ij} adjustment factors[1] increase the usefulness of the basic model by allowing for variations in 15 project attributes. Each factor allows estimators to account for different project attribute characteristics and thus fine tune the estimate. Estimators choose a value for each factor from a set of possible values ranging from slightly below one to slightly above one and then multiply the value by the values for the other 14 adjustment factors to produce a single Effort Adjustment Factor (EAF). The 15 factors are briefly described in Table 2.7. Estimators choose which value to multiply into the EAF based upon their assessment of the attribute's effect on the project. Estimators choose a value for each factor below, near, or above 1.0 depending on whether that particular attribute reduces, preserves, or inflates the basic effort.

[1] Adjustment factors are also known as cost driver attributes (Boehm:115), effort multipliers, effort adjustment factors, and cost driver multipliers (Gulezian:237). They all have a common theme -- inputs.

Table 2.7

Adjustment Attributes for Intermediate COCOMO Model (Boehm:118)

Area	Factor	Description	Multiplication Range
Product	RELY	Required Software Reliability	0.75 - 1.40
	DATA	Data Base Size	0.94 - 1.16
	CPLX	Product Complexity	0.70 - 1.65
Computer	TIME	Execution Time Constraint	1.00 - 1.66
	STOR	Main Storage Constraint	1.00 - 1.56
	VIRT	Virtual Machine Volatility	0.87 - 1.30
	TURN	Computer Turnaround Time	0.87 - 1.15
Personnel	ACAP	Analyst Capability	1.46 - 0.71
	AEXP	Applications Experience	1.46 - 0.71
	PCAP	Programmer Capability	1.42 - 0.70
	VEXP	Virtual Machine Experience	1.21 - 0.90
	LEXP	Programming Language Experience	1.14 - 0.95
Project	MODP	Modern Programming Practices	1.24 - 0.82
	TOOL	Use of Software Tools	1.24 - 0.83
	SCED[1]	Required Development Schedule	1.23 - 1.00 - 1.10

Table 2.8 shows an example of the possible values for the programmer capability (PCAP) attribute. As the programmer rating falls below nominal, the effort multiplier increases. The opposite case is also true. For cases which fall between two definitions, estimators can

[1] SCED has two multiplication ranges. A factor of 1.0 represents projects with a nominal schedule. Reducing the schedule from nominal results in a multiplication range of 1.00 - 1.23. Lengthening the schedule from nominal results in a multiplication range of 1.00 - 1.10. In both cases, successively larger multipliers are used as the schedule departs from the nominal estimate.

interpolate to find the appropriate multiplier. Thus, a 45th percentile programmer could have a 1.08 multiplier.

Table 2.8

Programmer Ratings and Effort Multipliers

PCAP Rating	Definition (Not based upon experience, only capability)	Multiplier
Very Low	Ranked at the 15th percentile of all programmers	1.42
Low	Ranked at the 35th percentile of all programmers	1.17
Nominal	Ranked at the 55th percentile of all programmers	1.00
High	Ranked at the 75th percentile of all programmers	0.86
Very High	Ranked at the 90th percentile of all programmers	0.70

Detailed Description. Third cut approximations require the most comprehensive form of COCOMO, the detailed COCOMO model. The detailed model increases estimation precision by breaking out all 15 adjustment factors into four phases of development: product design, detailed design, code/unit test, and integration/test (Boehm:364). Although the application of adjustment factors to separate phases leads to a more accurate description of an actual development process, it increases the predictive accuracy by only 2% (Boehm:521). This small increase in estimation accuracy is usually not worth the added expense of assigning adjustment factors to each phase. Thus, the detailed level of the COCOMO model is not used in general practice and most estimators use the intermediate form (Gulezian:237).

COCOMO Support. The COCOMO model normally is used to estimate the development effort of a software project (REVIC:4). But almost all the attribute factors which affect the development process also affect the support process. The model needs only a few adjustments to switch from predicting development effort to predicting support effort. The main adjustment required is scaling the man-months of effort. A prediction for development forecasts the entire effort for the development phase. However, support effort is usually for a fixed period of time, typically near one year. In a sense, the support model scales a total development effort since scaling is done proportionally. Boehm calls this scaling, which is the expected percentage of code changed for a specific period of time during support, the Annual Change Traffic (ACT) (Boehm:536). For example, if a development effort prediction was 120 man-months and the ACT was 10%, the expected support effort for one year would be 12 man-months.

Another support adjustment to the COCOMO model involves altering three of the adjustment factors shown in Table 2.7: SCED, RELY, and MODP. The required development schedule (SCED) is only a factor during development and is dropped as an input for support estimation (Boehm:129). The next adjustment factor, required software reliability (RELY), measures the probability that the software will perform its intended functions satisfactorily over its next run or its next quantum of

execution time (Boehm:372). Table 2.9 shows how the RELY values change from development to support.

Table 2.9

RELY Ratings (Boehm:374,538)

Phase	Ratings				
	Very Low	Low	Nominal	High	Very High
Development	0.75	0.88	1.00	1.15	1.40
Support	1.35	1.15	1.00	0.98	1.10

Table 2.9 quantifies the fact that programming low reliability software during development requires less effort, but supporting low reliability software requires more effort. The final altered adjustment factor, modern programming practices (MODP), also manifests different effects between development and support except, in this case, modern programming practices help reduce effort in both development and support. The difference is in the degree of effect as shown in Table 2.10.

Table 2.10

MODP Ratings (Boehm:538)

Phase		Ratings				
		Very Low	Low	Nominal	High	Very High
Development		1.24	1.10	1.00	0.91	0.82
Support (Size)	2K	1.25	1.12	1.00	0.90	0.81
	8K	1.30	1.14	1.00	0.88	0.77
	32K	1.35	1.16	1.00	0.86	0.74
	128K	1.40	1.18	1.00	0.85	0.72
	512K	1.45	1.20	1.00	0.84	0.70

For small sized changes, the support MODP ratings are very close to development MODP ratings. However, as the change size increases, the impact of MODP increases in both a positive and negative manner. Thus, as the size of the support change increases, the risk and payoff of using MODP increases.

The previous COCOMO descriptions show that COCOMO has three modes and levels of detail. Those simple options along with the minor alterations for the support phase make COCOMO a relatively simple and usable model. While REVIC uses all three modes, the intermediate level of detail, and the support alterations, COCOMO has yet another characteristic that makes it usable; COCOMO is straightforward to calibrate.

Calibration. Calibration of any model can improve the accuracy of its prediction. In fact, Thibodeau, in his evaluation of software cost

estimating models for the Rome Air Development Center, showed that the calibration of model parameters may be as important as the structure of the model in explaining estimating accuracy (Thibodeau:6-6). If the prerequisite data is available, all three levels of the COCOMO model are straightforward to calibrate. The intermediate COCOMO model can be calibrated in three ways depending upon the depth of calibration. All three calibrations adjust some parameter(s) of the COCOMO model through least squares regression to adjust for new or altered historical data. The result should be an improved model which predicts better for those efforts closely resembling the historical data used for calibration. The first calibration method calibrates only the coefficient of the basic model (see equations 2.7 - 2.10). Boehm recommends this method if the project's historical data has less than ten data points (Boehm:529). Calibrating the exponent of COCOMO equations with less than ten data points may unduly bias the exponent because one erroneous data point may overwhelm the contribution of the other nine data points. The least squares regression equations for both coefficient and/or exponent calibration are shown in equations 2.12 through 2.15.

$$Q_i=(KDSI_i)^b\Pi_i \qquad \text{Eq. 2.12}$$

$$a=\frac{\sum_{i=1}^{n} MM_iQ_i}{\sum_{i=1}^{n} Q_i^2} \qquad \text{Eq. 2.13}$$

where i = 1, 2, ... the number of historical data points
Π = product of C_{ij} adjustment factors for a data point
a = coefficient for a set of data
b = exponent for a set of data (given)
Q_i = temporary variable from equation 2.12
MM_i = man-month

If more than ten historical data points are available, then the coefficient and the exponent can be calibrated at the same time. The resulting calibration equations are still least squares regressions but include the log function to account for the exponent and are solved simultaneously.

$$a_0=n$$

$$a_1=\sum_{i=1}^{n}\log(KDSI_i)$$

$$a_2=\sum_{i=1}^{n}[\log(KDSI_i)]^2 \qquad \text{Eq. 2.14}$$

$$d_0=\sum_{i=1}^{n}\log(MM/\Pi)_i$$

$$d_1=\sum_{i=1}^{n}\log(MM/\Pi)_i\log(KDSI)_i$$

where a_0, a_1, a_2, d_0, d_1 = intermediate variables for Eq. 2.15
n = number of historical data points
i = 1, 2, ... the number of historical data points

$$\log(a) = \frac{a_2 d_0 - a_1 d_1}{a_0 a_2 - a_1^2}$$

$$b = \frac{a_0 d_1 - a_1 d_0}{a_0 a_2 - a_1^2} \qquad \text{Eq. 2.15}$$

where a_0, a_1, a_2, d_0, d_1 = intermediate variables from Eq. 2.14
a = coefficient for a set of data
b = exponent for a set of data

The third calibration method adjusts the coefficient, exponent, and all the effort multipliers for each effort adjustment factor and each mode. In this calibration method, Gulezian extends the least squares regression procedure to the multivariable case and improves the estimating accuracy of COCOMO. For Boehm's original database (Boehm:496), the intermediate form of COCOMO predicts within 20% of the actual effort 68% of the time while Gulezian has improved the intermediate model to predict within 20% of the actual effort 75% of the time (Gulezian:240). The extra accuracy does require more mathematical manipulations during calibration and would likely involve using a personal computer, but the extra accuracy is available.

COCOMO Strengths. The choice of modes, levels of detail, and methods of calibration point to one of the advantages that COCOMO has over other models: COCOMO is flexible. Estimators have had access to the model and have studied the model for over ten years (Gulezian:236). This open model permits estimators to explore inner workings, avoid pitfalls, and adapt the model to different situations. The key element of

model flexibility is the ability to change the input parameters and environmental factors. The parameters and factors which define the model are not magically pulled from a hat. They are carefully derived through statistical inference. If the environment of the software project changes severely, then estimators can rederive the parameters and adjustment factors to produce a model which more accurately reflects the current environment. This process is called calibrating the model. A properly calibrated model has been shown to give much more accurate results than an uncalibrated or badly calibrated model (REVIC:4).

Another advantage of COCOMO is its wide-spread use. For example, every DOD Defense Plant Representative Office (DPRO) has used the COCOMO model to help estimate software (ENREV:1). This commonality helps organizations produce results which are understandable and applicable to other organizations. Since many organizations understand the environmental factors that affect an effort estimate, the organizations can speak the same estimation language.

A final advantage of the COCOMO model is its usability. Dr. Barry Boehm, the creator of the COCOMO model, fully explains the implementation of the model is his book *Software Engineering Economics* (Boehm:Chap 5 - 9, 23 - 30). If estimators encounter difficulties when implementing the model, they can refer directly to the expert's handbook. Furthermore, since the model is simple to program on a computer, software versions of the model which run on the common personal

computer are available. The Enhanced REVIC Advisor (ENREV), developed by Ernst, tutors users through each step of the estimation process and explains all the terms needed for input (ENREV:12).

COCOMO Weaknesses. Of course, the COCOMO model is not the panacea for estimators' problems. The model does have weaknesses. Although calibration was mentioned as a strength, it is also a weakness. While calibration offers flexibility to change environments, it is mandatory when environments change significantly. The model becomes inaccurate unless it is calibrated to the new environment (van Genuchten:38) and calibrating the model is not trivial. First, input data concerning the two basic equation parameters and the 15 adjustment factors must be available. This requires enough data to establish trends for each input so that new values can be inferred. This kind of data is scarce and, if found, is usually of poor quality. Secondly, someone with statistical skill and COCOMO knowledge must actually perform the calibration. Finding a qualified estimator to calibrate a model can be difficult. The lack of these prerequisites often constrain estimators to use the latest model on hand regardless of the model's proper calibration.

Another weakness of the COCOMO model is the lack of specific adjustment factors for the support phase of a project. Many development attributes do indeed set support attributes, but to expect that support is merely an extension of development can mislead estimators about the true nature of the support environment. Some support factors, such as support

growth phases or block changes, are unique to the support phase. COCOMO does not sufficiently provide for these support-unique inputs (Ferens1:8,9).

The greatest weakness of the COCOMO model is the model's reliance upon KDSI as a primary input. First of all, *KDSI* is rarely estimable early in a software life-cycle (Bourque:161). But fortunately, as the support phase nears, actual *KDSI* values can be input into a support model. Also, the definition of a delivered source instruction (DSI) is open to some interpretation. Boehm defines a DSI as a single, delivered line of code which excludes comments but includes job control language, format statements, and data declarations (Boehm:59). This definition reduces misinterpretation but does not eliminate it. Unless estimators follow the explicit definition used during calibration, they will bias their results. Finally, one DSI for a particular programming language is not necessarily equivalent to a DSI for another programming language (Lehner:139). As a result, simply switching programming languages can bias effort prediction. These three weaknesses of mandatory calibration, missing support adjustment factors, and unreliable *KDSI* inputs often compel estimators to use or create other models.

COCOMO & REVIC. As mentioned earlier, REVIC is an implementation of the COCOMO model. Maj. Ray Kile programmed REVIC, a military version of COCOMO, using a data base of 281 completed contracts from the Rome Air Development Center database

(ENREV:1) to calibrate a new set of coefficients and exponents for all three modes of the model. REVIC also includes an additional mode, with its own coefficient and exponent, for Ada developments using an object-oriented analysis (REVIC:2). Also, REVIC also has 4 extra adjustment factors: requirements volatility, required reusability, classified security application, and management reserve for risk. To further aid estimators, REVIC includes a simple sizing model which sums the sizes of separate computer software components (CSCs). Users can provide a low, high, and most probable size for each CSC. These CSC size ranges can help set a total size range. The extra descriptors combined with a simple interface have made REVIC a popular model throughout DOD.

Conclusion

This literature review examined definitions of software support as well as the processes used to accomplish support. The essential areas of a capable estimating model including size, maintainability, productivity, and schedule were explained. Finally, the COCOMO model was scrutinized to provide an understanding of its algorithms, strengths, and weaknesses. The sources reviewed all indicate that the existing software maintenance effort estimation tools are inadequate for estimating software support. In general, the tools are inaccurate or inappropriate. However, many drawbacks of existing models have been addressed in the literature along with possible improvements. Combining some of the improvements and

addressing some of the drawbacks, provides a potential for improving support cost models.

III. Methodology

Chapter Overview

This chapter details the steps needed to analyze the data to support each of the hypotheses. For the first hypothesis, the functional bottom-up construction of a software cost estimation model, the method used to model the various levels of the block change process is described. Also described is the process used to develop an estimation model from the block change process model. Since no data was available to validate the new model, a discussion of how validation can be accomplished is presented. For the second hypothesis, the functional top-down calibration of a statistically-*based software effort estimation model, discussion is* limited to the data collection method and a description of the data originally sought. Further work on this hypothesis was impossible due to lack of data. However, a method for formally evaluating model performance once data becomes available is presented. This discussion follows a chronological flow of events from the data input sequence to applying statistical rejection criteria.

Hypothesis 1 Methodology

Block Change Model Methodology. Documenting the block change process as a model was a necessary first step to program a software support cost model. The objective was to create a cost model that is closer

to the detailed estimate methodology than current parametric models. The transition from paper model to software design is eased by the proper selection of documentation style. This research uses the Object Oriented Design documentation style as described by James Rumbaugh and others in *Object-Oriented Modeling and Design* because of its versatility and prior experience with that design method. Selecting a software design methodology to document the real block change process has two main advantages. The first is the reduction in software system design time gained by avoiding translation of requirements from some other abstraction media into a software design. The second is direct traceability from the model to the software design. Both of these advantages resulted in fewer errors in the final product.

Rumbaugh's design method revolves around three model types: the object model, the functional model, and the dynamic model (Rumbaugh:6). The model types describe different characteristics of the problem space, and, when combined, form an overall description of a system. Each model type is described in the following paragraphs.

The object model is the primary description of the agents within the process. It is described by Rumbaugh as a graphic representation that "captures the static structure of the system by showing the objects in the system, the relationships between objects, and the attributes and operations that characterize each class of objects" (Rumbaugh:21). An

object is defined as "a concept, abstraction or thing with crisp boundaries and meaning for the problem at hand" (Rumbaugh:21). No process or temporal information on the system is captured by the object model.

The functional model shows how data flows from one process to another in a series of data flow diagrams (Rumbaugh:123). It traces the inputs of the system through transformational processes within the system to the outputs of the system. No information on the static structure or the temporal behavior of the system is captured by the functional model.

The dynamic model describes the temporal behavior of the system. It uses state transition diagrams to describe "the sequences of operations that occur in response to external stimuli" (Rumbaugh:84). No structure or process information is captured by the dynamic model.

The importance of each model depends entirely on the system being modeled. For example, if the system doesn't change over time, the dynamic model is trivial and may be omitted. The purpose of each model is to force an analyst to examine and understand the static structural, procedural, and dynamic facets of the system. Each model uses a standard set of labeled symbols to pictorially describe system behavior. (A summary of each model's notation is shown in Appendix A.) Label definitions for all the models are contained in a data dictionary. The data dictionary coupled with the graphic nature of the models allows the analyst to interact with system experts to avoid miscommunication.

Model Design Process.

Step 1. The first step captured the attributes of a single iteration of a block change process as described in the literature review. The description was in the form of an object model, a functional model, a dynamic model, and a data dictionary. The object and functional models provided a complete picture of the block change process, so the dynamic model was dropped. The object and functional models were verified by consulting with HQ/AFMC and by comparing the models to the available literature.

Step 2. The second step started with the design from the models of a single block change process and evolved into the models of a series of block change processes. This "ideal" (described in Chapter II) support cost estimation model for the block change process was also documented in the form of an object model, a dynamic model, a functional model and data dictionary. The result of this step was three graphic representations that combine to form a description of a software support cost estimation model. At this point the ideal model was still generic. Any development estimation tool could, in theory, have been adapted to emulate the ideal model by repeated development estimates or by recoding. The ideal model was verified by comparing the object, functional, and dynamic models with the Step 1 models and with available literature.

Step 3. The third step instantiated the ideal model based on the COCOMO equations discussed in Chapter II. This instantiation

placed the COCOMO mathematical models into the ideal model's object, functional, and dynamic representations. Verification consisted of comparing the instantiation with the ideal model and ensuring the COCOMO mathematical model was correctly transcribed.

Step 4. The fourth step was to prototype the instantiation. Visual Basic was selected as the coding language since it is a quick, easy, object-centered language designed to create code for the Windows environment. Rudimentary verification was obtained by comparing the estimate of a single block change of all new code to a REVIC estimate of a development effort with identical parameters. The support costs were compared with other models to determine the magnitude difference between Air Force recommended models and the new model. The results are presented in Chapter IV.

Completion of the above steps did not result in a calibrated or validated model, only a feasible model. Calibration and validation required actual time series support data that were not available. The result of calibration and validation would have been an expected prediction accuracy. Without these steps, the model has an unknown accuracy. However, validation can be accomplished by calibrating the model and gathering the statistics discussed under Model Comparison Methodology later in this chapter.

Hypothesis 2 Methodology

Data Collection. Data could not be obtained to examine the effect of functional calibration on model accuracy. (For a full account of the problems encountered, please refer to Chapter IV.) The following paragraphs discuss what type of data needed to be collected and why.

Historical data from the software block change process was needed from as many combat aircraft as possible. As a minimum, for each block change, three pieces of data were needed: the size of the effort (divided into new, modified, and deleted lines) measured in thousands of lines of delivered code (KDSI), the magnitude of the effort to produce the code measured in man-months (MM), and a functional description to place the data into one of our proposed functional categories.

Also needed were some limited data concerning the detailed characteristics of each block change to help fine tune a model's predictive capability. The data collection forms prepared to obtain this data are shown in Appendix B. However, the data were not available. Since the research assumes that KDSI is the single best predictor of effort, the inability to obtain detailed characteristics (other than size) should not hinder the ability to determine whether functional division of block changes improves predictive accuracy. Extra block change characteristics could help determine to what degree functional division improves accuracy for each particular model. Collection of extra descriptive data suitable for each model was simply impractical due to time and money constraints.

Therefore, the potential benefit of setting the production characteristic input variables was balanced against the potential error (and effort required) which might be introduced by choosing inappropriate values, and a decision was made to use the nominal values provided in the model for those inputs.

Attempts were made to obtain historical block change data from two basic sources. The first source was existing Air Force databases which have already compiled the input values and output values concerning this research. As stated in Chapter I, many prediction models are built by collection and statistical regression of historical data. It seemed natural that historical data would be systematically collected and stored to help make new models or improve existing ones. While the Air Force recognizes the need for historical data, the current databases were found to contain data for software development but not software support. Because of this lack of support data, questionable accuracies (Ourada:4.1), and unusable variable formats of the database, no useful data was obtained from this source.

It was expected that more and higher quality block change data could be obtained by contacting the System Project Offices (SPOs) located at Wright-Patterson AFB (WPAFB). Virtually every Air Force combat aircraft is represented by some SPO located on the base. The SPOs assisted in the collection of some data by providing contacts within the appropriate software maintenance organization (SMO). In all cases, the

SMO was at an Air Logistic Center (ALC) and the ALCs provided all the data found.

Model Comparison Methodology

This section describes how to recalibrate existing (and future) support cost models and gauge the resulting accuracy increase. This methodology is generic and can be used with any software cost model once data becomes available. The data requirements are discussed under Hypothesis 2 Methodology. Recall from Chapter II that ten data points or more are required for accurate calibration of the REVIC exponent. Ten or more data points should suffice as a minimum for other models as well. Model calibration should be done according to the User's Manual for the software. Once the calibration procedure is complete the model is calibrated to the new data set and will be ready for estimating.

Data Input Sequence. If data had been available, the models would have been operated under three different scenarios. The first scenario was to input no calibration data at all. In other words, this scenario did not include recalibrating the model. The model was to be run "out of the box" using its existing calibration. The predictions obtained from this scenario would serve as the baseline to determine the predictive improvements of each subsequent scenario. The second scenario included recalibrating the models based upon all the block change data from all the functional categories. Although this scenario does not functionally calibrate the

models according to categories chosen in Chapter I, this calibration should produce a model that is functionally narrowed to the area of OFP software. The models calibrated in this scenario were expected to predict better than the uncalibrated models under the first scenario. The third scenario included recalibrating the models based upon the inputs from the functionally stratified data. The models calibrated according to the third scenario were expected to predict the best.

Apply Statistics. Statistical techniques could be used to evaluate which scenario produces the best predicting model. Devore defines a statistic to be any function of random variables constituting one or more samples, provided the function does not depend on any unknown parameter values (Devore:231). The two major constructs of a statistic are a set of random data and a function to operate on the random data. From this basic starting point, boundaries on the methodology begin to emerge. First, the statistical functions chosen to employ as an evaluating tool center around the branch of statistics known as analysis of variance (ANOVA). The nature of this research fits this type of analysis very well. As the predictive capability of a model increases, the variance of the error of the model's prediction will decrease. The predictive accuracy of a model is measured by the variance of the errors of that model. Second, the limited available data could not provide random data. Ideally, a random sample would be chosen from data on all the software support efforts in

the entire Air Force history. Limited data forced the assumption that the data collected would not significantly bias the results.

One way to help compensate for this weakness is to apply more than one statistical function to the available data. Four statistics, three of which are related to ANOVA techniques, can be applied. The first statistic is the adjusted coefficient of multiple determination (R^2). The value of this statistic can be interpreted as the proportion of the total variation of the observed values[1] that can be explained by a multiple regression model and adjusted for the degrees of freedom[2] (Neter: 241). The equation used to calibrate R^2 is shown in equation 3.1.

$$R^2 = 1 - \frac{\frac{SSE}{n-p}}{\frac{SSTO}{n-1}} = 1 - \frac{\sum_{i=1}^{n} (E_{act_i} - E_{est_i})^2}{\sum_{i=1}^{n} (E_{act_i} - E_{mean})^2} * \frac{n-1}{n-p} \qquad \text{Eq. 3.1}$$

where
R^2 = coefficient of multiple determination
SSE = sum of squares for errors
$SSTO$ = sum of squares total
n = number of data points
p = number of estimated parameters (2)
i = 1, 2, ... n
E_{act} = actual effort
E_{est} = estimated effort
E_{avg} = average effort

---

[1] An observed value for this research would be the effort in man-months associated with a particular block change.

[2] The degrees of freedom can be considered as to the number of observations minus the number of parameters required to be estimated. Smaller sample sizes have smaller degrees of freedom.

In equation 3.1, *SSE* is associated with the variation of the estimation errors of the regression model, *SSTO* is associated with the total variation of the observed values, n is the number of observations, and p is the number of parameters estimated by the regression model. The possible values for R^2 range from 0.0 to 1.0. The variation explained by the regression is assumed to come from a linear regression of the historical data. If the assumed regression is not linear, then the regression must be transformed into a linear regression or a different statistic should be used. For example, COCOMO uses an exponential model that can be transformed into a linear model by taking the logarithm of both sides of the model equation.

The second statistic is the relative root mean square error (*RRMS*) where error is the difference between the actual and estimated value. The statistic is the ratio of the root mean square over the average actual value (see equation 3.2).

$$RRMS = \frac{RMS}{E_{avg}} = \frac{\sqrt{\frac{1}{n}\sum_{i=1}^{n}(E_{act_i} - E_{est_i})^2}}{\frac{1}{n}\sum_{i=1}^{n}E_{act_i}} \qquad \text{Eq. 3.2}$$

where
$RRMS$ = relative root mean square
RMS = root mean square
n = number of data points
i = 1, 2, ... n
E_{avg} = average effort
E_{act} = actual effort
E_{est} = estimated effort

The third statistic is a simple prediction level. The function is the ratio of the number of samples, k, whose magnitude of relative error (*MRE*) is less than or equal to a given percentage, l, over the total number of samples, n (see equations 3.3 and 3.4). Although the value of the statistic is simply a ratio of samples which meet the minimum *MRE*, the statistic is easy to comprehend and does provide a broad gauge to measure an improvement. The possible values for the prediction function range from 0.0 to 1.0. A minimum value for *MRE* of 25%, where *MRE* is computed as in equation 3.4, was used in this study.

$$\text{PRED}(l) = \frac{k}{n} \qquad \text{Eq. 3.3}$$

where l = prediction level percentage
k = number of samples within prediction level
n = number of samples

$$MRE = \frac{|E_{act} - E_{est}|}{|E_{act}|} \qquad \text{Eq. 3.4}$$

where *MRE* = magnitude of relative error
E_{act} = actual effort
E_{est} = estimated effort

The last statistic is the proportion of sample estimates which are more accurate after functional calibration. By applying the above statistics to each model before and after a calibration, the predictive value can be assessed. If the predictive value of a particular block change sample improves, then it is assigned a value of one. Otherwise the block

change sample is assigned a value of zero. The sum of those samples which improve after calibration can be described as a binomial variable. The proportion is the value of the binomial variable divided by the total number of samples. If the functional orientation of the model does nothing to improve the predictive accuracy of the model, then an improvement ratio near 0.0 is expected. A ratio near 1.0 would indicate improved predictive accuracy.

Apply Rejection Criteria. For each statistic, a predetermined rejection criteria is needed to determine whether the value of the statistic supports or rejects the hypotheses. The following paragraphs provide recommendations and a discussion of alternatives and potential problems. While the choice of rejection values is somewhat arbitrary, a pass-fail line is needed to judge any improvement in the predictive capability of functionally oriented models.

For the adjusted coefficient of determination, an R^2 value of .9 or higher indicates that the predictive capability of that model is acceptable. The R^2 associated with a given model can be used to compare different models to determine which explains more variation. However, since detailed input information required for each model may not be available, comparing the R^2 of one model with high quality inputs to the R^2 of another model with low quality inputs may not be conclusive. These inappropriate comparisons may restrict the ability to state which model is better than the others. When trying to support the main hypothesis of

improvement, the value of R^2 will either rise or not rise. By observing the change in R^2, an assessment of a model's improvement can be made.

For the root mean square error, an *RRMS* value of 0.25 or less to indicate an acceptable model was chosen. Using this statistic to compare two models may be inconclusive because of different input requirements for each model. However, the statistic can be used to determine whether an individual model shows improvement after functional orientation.

For the percentage prediction function, a predictive percentage of within 25% of the actual value on 75% of the samples was chosen to indicate an acceptable model. Once again, using this statistic to compare two models may be inconclusive because of different input requirements for each model. However, the percentage in the prediction zone can also be used to determine whether an individual model shows improvement after functional orientation.

For the improvement ratio, the hypothesis that the model does show improvement from functional orientation is accepted if the ratio supports a 0.1 level of significance. This rejection criteria is the main criteria for the hypothesis that functionally calibrating a model improves predictive accuracy.

Conclusion

This chapter described the methodology for constructing and evaluating the two hypotheses. The test tool of the first hypothesis is a

model which competes with the four Air Force recommended models. The design process for this new model and the calibration/verification methodology were presented. Under the second hypothesis, four statistics that could be used to evaluate the four Air Force recommended models within the five functional categories were reviewed. However, the second hypothesis could not be statistically tested due to a lack of useable data. Suggestions for gathering data to validate this hypothesis in future research are presented in Chapter V.

IV. Findings

Chapter Overview

This chapter summarizes the activities for evaluating the two hypotheses. First, in support of the bottom-up model construction (Hypothesis 1), Boehm's COCOMO maintenance equations are adapted to the avionics PDSS process to produce a prototype software model. Second, in support of our functional calibration (Hypothesis 2), calibration procedures are defined for the prototype model in order to represent a set of historical data. Next, attempts to obtain useful support data from Air Force organizations are reviewed and then, because of a lack of useful data, fictional data is generated as a substitute. Finally, the fictional data is processed through the prototype model to demonstrate how the results of processing actual data might appear. The result of the model building was a software tool that could confirm or deny the hypotheses with actual data.

Hypothesis 1: Functional Construction of a Software Estimation Model

Constructing the prototype model occurred in three phases. The first phase included describing an avionics block change process in object oriented terms. The second phase consisted of a review of the description to determine if it adequately reflected the existing software support environment. After validation the description, the third phase involved

implementing parts of the description in Visual Basic[1] code. (See Appendix C for the source code.) The Visual Basic program was a tool for demonstrating our hypotheses and a prototype for constructing a software support estimation program.

Model Design. Using the object oriented design methodology described in Chapter III to document the block change process was very successful. To begin, we modeled a single block change cycle as an object model, a functional model, and a dynamic model. The dynamic model provided little insight into the block change process; it tracked only whether software was currently in a block change process or not. Therefore, the dynamic model was removed from the model set. The set was now composed of an object model that identified the important players in the block change process and a functional model that described the block change process as a series of subprocesses. The final results are detailed in Appendix D and summarized in Figure 4.1.

[1] Visual Basic is a registered trademark of the Microsoft Corporation.

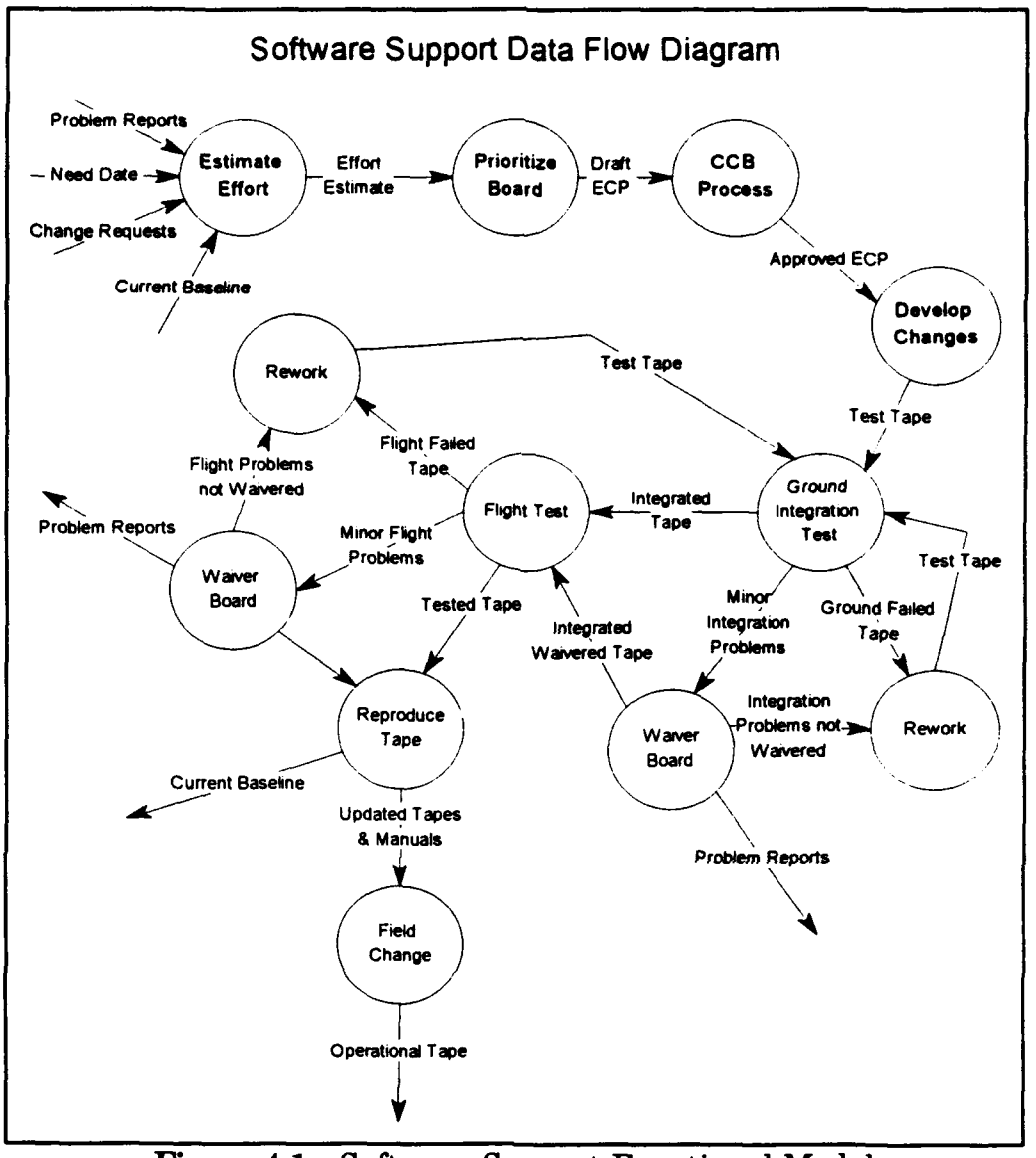


Figure 4.1 - Software Support Functional Model

Model Design Validation. Our model diagrams served as a common reference for comparisons with support models from existing avionics block change process. AFMC/HQ recommended discussing the support process with a process action team (PAT) at Sacramento ALC who had completed a detailed review of the entire PDSS process in October 1990. The PAT compiled a detailed report using a computer aided software engineering

(CASE) design tool and structured system design techniques to document a PDSS process (Talbot). Structured design, a predecessor to object oriented design, was compatible to our model for direct comparison. Comparing the structured analysis data flow diagrams prepared by the PDSS PAT (see Figure 4.2) to our object oriented functional models (see Figure 4.1), which used the same notation, was direct and simple. The models correlates well with only a few minor differences.

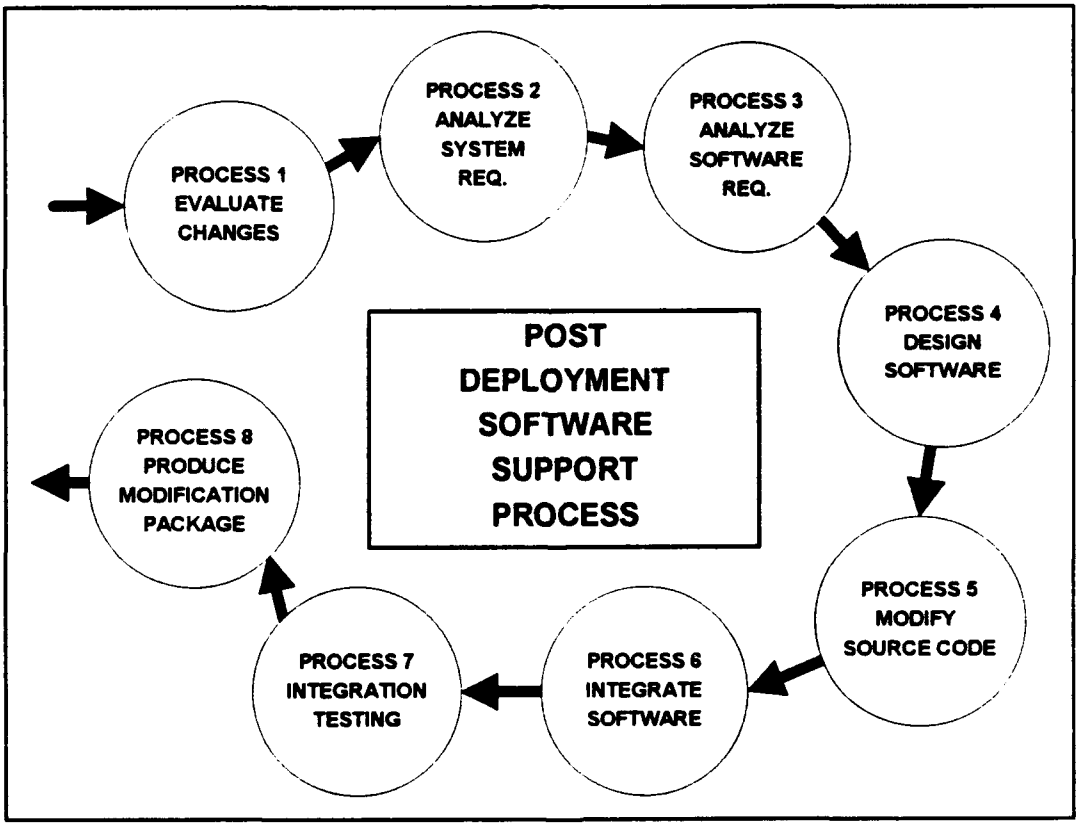


Figure 4.2 - PAT Functional Model (Talbot:15)

We contacted Sacramento ALC to reconcile the differences between the PDSS PAT model and our block change model. The differences stem

from an assumption made by the PAT; their model assumed successful completion of all software support activities on the first try. The PAT model did not show any rework or waiver paths. Since the block change model had no assumption of success, it included rework and waivers as alternative paths from both the integration and flight test processes (see Figure 4.1). Table 4.1 is a comparison of the single block change cycle model to the PAT model and the MIL-HDBK-347 process description. The close correlation of the our model to the PAT model was tacit confirmation of our model.

Table 4.1

Block Change Process Model Comparison

HYPOTHESIS 1 Block Change Process Model	SACRAMENTO ALC PAT Process	MIL-HDBK-347 Process
1) Estimate Effort	1) Evaluate Changes	1) Initial Analysis
2) Prioritize Board		
3) CCB Process		
4) Develop Changes	2) Analyze System Requirement	2) Develop Software
	3) Analyze Software Requirement	
	4) Design Software	
	5) Modify Source Code	
	6) Integrate Software	3) System Integration and Test
5) Ground Integration Test	7) Integration Testing	
6) Flight Test		
7) Waiver Board	(Not Modeled)	(Not Modeled)
8) Rework	(Not Modeled)	(Not Modeled)
9) Reproduce Tape	8) Produce Modification Package	4) Product Logistics
10) Field Change		

Model Construction.

Choosing a Language. Confirming the block change support model signaled the clearance to translate the model description into software code. We chose to code our model in the Visual Basic programming language for several reasons. The foremost reason for choosing Visual Basic was the simple and understandable language syntax. The mathematical algorithms implementing the support model

were easy to comprehend because of the simple syntax. Another reason for selecting Visual Basic was the availability of pre-written software components, called controls, which eased our programming task. These controls also provided an excellent prototyping environment for quickly trying different possibilities. The last reason for choosing Visual Basic was the supplemental software included with the Visual Basic package which had the capability to create executable and distributable programs. Thus, anyone who wished to run our prototype could obtain free copies. The Visual Basic language proved to be a capable and flexible language for prototyping.

Refining the Model. Our intention of programming with Visual Basic was not to program the entire PDSS process but to program only that portion of the process needed to test the hypotheses. The basic approach was to adapt the maintenance portion of Boehm's COCOMO model to reflect block change characteristics cited in the literature review and the PDSS process as modeled in Figure 4.1. The model accepted a development code size, an expected change in code size, and a support attributes as a starting point to iterate several successive block changes. While Boehm's COCOMO model used annual change traffic (ACT) to represent the expected change size, the prototype introduced another size measurement called block change traffic (BCT). Furthermore, while the COCOMO model did not explicitly vary any attributes during successive

block changes, the prototype model altered some attributes to account for temporal changes from block change to block change.

Block Change Size. As stated in Chapter II, Boehm described the expected change in code size during one year of support activity as the ratio of the expected number of added and modified delivered source instructions (DSI) over the total number of DSI. This ratio, called annual change traffic (ACT), was the predecessor to the prototype ratio, BCT. BCT was the ratio of equivalent DSI (EDSI) over the total number of DSI. Note that while the prototype EDSI was conceptually the same as EDSI from Boehm's conversion cost estimating (both are an equivalent size and are weighted), the prototype EDSI was not calculated in the same manner. The model did not use Boehm's conversion cost estimating, so all future references to EDSI mean the prototype EDSI, not Boehm's. The model EDSI was the weighted sum of the expected number of added, modified, and deleted DSI for a particular block change. While Boehm's ACT did not include deleted DSI or weight any code, the model EDSI included lines of deleted code and weighted the three change categories. Weighing the expected added, modified, and deleted lines of code consisted of multiplying each change category by values from SOFTCOST-R's reused code weightings in Table 2.2. Since the prototype did not distinguish differences between modules and lines, it added the weightings for module and line deleted and for modules and lines modified or changed. The prototype model also ignored the retested modules because avionics testing

effort was better estimated by other means. The resulting equation for EDSI is shown in equation 4.1.

$$EDSI = 0.53 * KDSI_{Added} + 0.51 * KDSI_{Modified} + 0.26 * KDSI_{Deleted} \quad \text{Eq. 4.1}$$

where $KDSI$ = thousands of delivered source instructions
$EDSI$ = equivalent $KDSI$

This definition of EDSI was an attempt to better define the block change size. Remember that the major correlation in predicting effort was between size and effort. As stated in Chapter II, better predictions in size estimates will produce better effort estimates. Although the code size during support was not exactly reused code as used by SOFTCOST-R, reused code was similar enough to redeveloped (support) code to warrant the reused weightings as a starting point for a better estimation of size.

Temporal Changes. With an adequate description of a block change size, we described the temporal effects for each block change. The front and back ends of the bath-tub curve, as described in Chapter II, can accompany successive block changes. These temporal effects could come from at least four potential sources: learning, size growth, memory and throughput utilization, and entropy. Each of these sources is discussed in the following paragraphs.

The first temporal effect encountered in the support phase of software was the learning effect. The learning effect was a direct result of support personnel working with software and its documentation. As these

personnel increased their knowledge of the support software, they could better identify errors and more efficiently correct them. However, the learning effect did not continue throughout the entire PDSS phase. According to literature cited in Chapter II, learning lasted from three to six years (SYSCON:44). A possible contributor to this time window is personnel turnover. When software personnel depart from a position, new personnel must replace them, but these new personnel are not as knowledgeable or experienced. The departing personnel take with them much of their learning, and that learning is lost.

The learning curve implies a relationship between the complexity of the software and the rate at which learning occurs; more complex software should take longer to learn. The prototype did not include a complexity/learning relationship because no literature sources were able to quantify it.

The prototype model did introduce a learning effect model by decreasing COCOMO's analyst experience (AEXP) and language experience (LEXP) multipliers for the first six block changes. Following the curve suggested by the SYSCON data in Table 2.5, the prototype multiplied the AEXP and LEXP attributes by values less than or equal to 1.0. The equation that calculated these values, equation 4.2, came from the averaged exponents of the design and development equations in Table 2.5 (see Appendix E for derivation). Equation 4.2 discarded the coefficients from Table 2.5 to force an initial multiplier of 1.0 for block change zero

(development). Succeeding block change multipliers were all less than 1.0, resulting in an effort reductions, and were applied equally to AEXP and LEXP for the first six block change cycles. In order to use the learning effect relationship, equation 4.2 assumed each block change was one year in length. A change to the block change duration would require additional calculations to maintain the documented relationship.

$$LE=(BC_i)^{-0.3745} \qquad \text{Eq. 4.2}$$

where LE = learning effect
BC_i = block change number
i = 1, 2, .. 6

The second temporal effect encountered in support was size growth in the software being maintained. As software was modified, it usually grew in size. Size was important to track because estimators needed to know how much of the target computer memory remained unused in order to estimate an increase in the support effort per line of code. A larger code size required more effort per line to support.

The prototype model encompassed a method to increase the size of the support code in a realistic manner. The term realistic means that the model complied with the constraint that size cannot grow beyond 100% memory capacity. Once software grows to over 95% of available memory, the model limited the size growth for successive block changes to one-half the remaining memory. Therefore, in later block changes, the software size grew very slowly and asymptotically towards 100%.

The specifics of size growth were implemented as a two part process. First, an estimator provided a development total code size and a memory utilization percentage. Next, the number of deleted and added lines of code for each block change were respectively subtracted and added to the previous block change size. The result was a changing total code size that typically increased in size since the number of added lines was usually greater than the number of deleted lines.

As the total size of the code increases, it can affect the next set of temporal effects, memory and throughput utilization. The prototype model treated these two effects as one because of their common characteristics. As the utilizations increase, they both increase the required support effort and they both are limited to 100%. Aircraft avionics systems have definite memory and throughput limits that can't be exceeded. As the memory and throughput limits are approached, support programmers must carefully engineer the code modifications to remain below the memory and throughput limits. When adding code near the memory and throughput limits, software support programmers must choose size efficiency over code simplicity and clarity. The resulting code is more complex than code at lower memory and throughput limits. This complex code requires greater intellectual investment and more time to create. The code is also more error prone. The prototype model included both temporal effects in two ways. The model set how the level of utilization affects one particular

block change and how the utilization level changes during sequential block changes.

For one particular block change, the levels of memory and throughput utilization could affect the original COCOMO attribute multipliers of the main storage constraint (STOR) and the execution time constraint (TIME) (see Table 2.7). Boehm defined the STOR rating as a percentage of main storage expected to be used by the subsystem and any other subsystems consuming the main storage resources (Boehm:410) and the TIME rating as the percentage of available execution time expected to be used by the subsystem and any other subsystems consuming the execution time resource (Boehm:401). Both of these definitions fit nicely with the objective of describing the effects of memory and throughput utilization[1]. Although Boehm already quantified the affects of these attributes with a set of four discrete multipliers, the prototype altered the development attributes with a continuous function based on the utilization percentage. The coefficients and exponents of the memory fill and timing fill relationships of the design and development phases from the SYSCON report (Table 2.3) were averaged to produce a composite relationship that covered support design and development. The resulting equations provided a multiplier to apply against the development STOR and TIME attributes as the memory and timing percentages varied during block cycle

[1] The prototype model considered available throughput as the mathematical inverse of available execution time. Both degrade at the same rate.

changes. Both equations 4.3 and 4.4 produced a multiplier of 1.0 at a utilization rate near 62% utilization, so the model used 65% utilization as the threshold for applying the equations. At utilization rates of 65% or more, the equations produced multipliers greater than 1.0 to represent an increased effort.

$$ME = 1.905 * (\%MemoryFill/100)^{1.425} \qquad \text{Eq. 4.3}$$

where ME = memory effect

$$TE = 1.82(\%Throughput/100)^{1.305} \qquad \text{Eq. 4.4}$$

where TE = throughput effect

At 95% memory and throughput utilization , both equations combined to yield a 3.01 multiplier. This 3.01 value implied that the effort required at 95% utilization rate is about three times greater than an effort at less than 65% utilization. This 3.01 value also compares favorably with the 95% result of 3.78 shown in Table 2.4.

Once the memory and throughput utilization effects were established, we determined how the utilization would vary from block change to block change. Because of an absence of historical time-series utilization rates for software support, the prototype substituted a simple rule: both memory and throughput utilization varied directly with size growth as described above. Although this rule applied to memory utilization, it did not precisely describe throughput utilization. For throughput, The model accepted this imprecision in order to keep the

model simple. However, since neither memory nor throughput utilization can exceed 100%, the prototype limited utilization growth to one-half the remaining utilization when utilization exceeded 95%.

Size growth, along with dwindling available memory and throughput, also contributed to the fourth bath-tub curve effect, design deterioration. We preferred the term design entropy although none of the literature reviewed used this term. The word entropy highlights an irreversible nature of design degradation as code is iterated through multiple block changes. We suspected design entropy was primarily a by-product of making adaptive changes to software. Later in the PDSS life cycle, as memory and throughput reserves dwindle, design entropy can be accelerated by the necessity to code for efficiency instead of simplicity. Efficient and complex code often leads to complex and degraded designs. Another contributor to design entropy could occur from poor software engineering practices during PDSS. Whatever the cause, the result was the same. The design lost flexibility, and the original structure of the software decayed as changing requirements are implemented. How quickly design entropy affected support depended on the original design. Simple, well-engineered, and well-documented code that is built and maintained with object oriented or structured design techniques should not show the effects of entropy as early as complex, poorly-designed code (spaghetti code). We faced a challenge trying to express and quantify this

concept. As a minimum, two components were needed - entropy timing and magnitude of impact.

With no data or literature quantifying the timing or magnitude of design entropy, we resisted the temptation to create a relationship out of thin air and, instead, left it out of the model. The model relied on the size growth and the reduction of available memory and throughput to capture the upward sloping portion of the bath-tub curve.

When all of the effects from this section were combined into one equation, the result was equation 4.5.

$$MM_{estimated} = (BCT)\, a(KDSI)^b \, (\Pi) * LE * ME * TE \qquad \text{Eq. 4.5}$$

where
MM = estimated man-months
BCT = block change traffic
a = calibration coefficient
b = calibration exponent
$KDSI$ = thousands of deliver sources lines of code
Π = product of the COCOMO multipliers
LE = learning effect
ME = memory effect
TE = throughput effect

Hypothesis 2: Functional Calibration to Improve Model Accuracy

Although equation 4.5 adequately described the general shape of a time-series of support efforts for a single unit of code, the coefficient parameter, a, and the exponent parameter, b, needed accurate values so the equation could produce the best estimates possible. Finding the best parameter values for a particular scenario was done through calibration from historical data. It was possible to calibrate equation 4.5 using the

calibration technique outlined in Chapter II (equations 2.14 and 2.15) by including BCT as shown in equation 4.6.

$$a(KDSI_i)^b = \frac{MM_i}{\Pi_i * BCT_i} = Q_i \qquad \text{Eq. 4.6}$$

where
a = coefficient parameter
b = exponent parameter
i = 1, 2, .. number of block changes
$KDSI$ = thousands of DSI total code size
MM = man-months of effort
BCT = block change traffic
Π = product of adjustment factors and temporal effects
Q = temporary variable for future equations

To begin the calibration process, the prototype model used a set of historical *MM*, *BCT*, and Π values to calculate a set of Q values. Each Q_i value represented an effort for correlation against a corresponding block change size. Although $KDSI_i$ and Q_i in equation 4.6 appeared to correlate total size and effort, the real correlation was between *EDSI* and effort. *BCT*, which was contained in *EDSI*, could be considered as a scaling factor to make Q_i appear as a typical development magnitude for the total development size. (Remember the MM_i value only included effort for a block change, not an entire development.) In order to correlate these points linearly (no exponents), equation 4.6 can be transformed with a logarithmic function to produce equation 4.7.

$$\log(a)+b*\log(KDSI_i)=\log(Q_i) \qquad \text{Eq. 4.7}$$

Equation 4.7, in the form of a line, permitted a least squares regression to a set of data points. The results of this regression were values for a and b which describe the line that best predicts a Q value (effort) given a *KDSI* value (size).

The continuation of the calibration process included building calibration equations similar to those from Chapter II (equation 2.14) to arrive at equation 4.8. (Remember that Q contains *BCT*.)

$$a_0=n$$

$$a_1=\sum_{i=1}^{n}\log(KDSI)_i$$

$$a_2=\sum_{i=1}^{n}[\log(KDSI_i)]^2 \qquad \text{Eq. 4.8}$$

$$d_0=\sum_{i=1}^{n}\log(Q_i)$$

$$d_1=\sum_{i=1}^{n}\log(Q_i)\log(KDSI)_i$$

where a_o = the number of data points (block changes)
a_1, a_2 = temporary variables for future equations
d_0, d_1 = temporary variables for future equations

Next, equation 2.15 can be applied to calculate a and b.

$$\log(a)=\frac{a_2d_0-a_1d_1}{a_0a_2-a_1^2} \qquad \text{Eq. 4.9}$$

$$b=\frac{a_2d_0-a_1d_0}{a_0a_2-a_1^2}$$

From a mathematical standpoint, equations 4.8 and 4.9 were similar to the COCOMO calibration equations from Chapter II and were straightforward to calculate. Keeping the exponential form of Boehm's equations, these equations captured a relationship between the size of the baseline code (*KDSI*) and the size of the change being made (*EDSI*). This relationship implied that supporting a given number of DSI in a large program required more effort than supporting the same number of DSI in a small program.

Interpreting the meaning of this support calibration depended on the data used to calibrate the model. If data was from a single PDSS lifecycle, then the calibration would capture the long-term, stable characteristics within that lifecycle. We labeled this case a horizontal calibration. During a development time frame, this type of calibration would be useful for predicting total lifecycle cost from equations calibrated to previous support lifecycles. During a support time frame, this type of calibration would be useful for predicting the next block change from equations calibrated to previous block changes. However, these calibration techniques might be inappropriate if the attribute adjustment factors did not follow a bath-tub curve. This last scenario may be better suited for the case we labeled as vertical calibration.

Vertical calibration uses data from the i^{th} block change cycle from different software products for calibration. This calibration could capture common characteristics across similar systems at the i^{th} block change and,

presumably, could be used for predicting the i^{th} block change cost of a similar system. Vertical calibration is equivalent to calibrating the development equation using development data because the historical data comes from different software systems. However, there is no developmental equivalent of horizontal calibration.

Which calibration works the best for a given scenario? There was insufficient time or data to experiment with horizontal versus vertical calibration. However, given sufficient data, this comparison could be accomplished using Hypothesis 2 (functional calibration of existing models) methodology and the model comparison methodology.

In an attempt to validate the calibration algorithm, we entered Boehm's database (Boehm:496) into the prototype and calibrated each mode. A re-derivation of the coefficient and exponent values for all three COCOMO modes was expected. However, as shown in Table 4.2, the results did not duplicate the coefficient or exponent for the organic and semi-detached modes. The embedded mode values were much closer.

Table 4.2

Calibration Comparison

Mode	Boehm Parameters		Prototype Parameters	
Organic	3.2	1.05	4.3000	0.9337
Semi-detached	3.0	1.12	3.1855	1.1033
Embedded	2.8	1.20	2.7838	1.2090
All data	–	–	2.8309	1.1581

One possible reason for the differences shown in Table 4.2 could be rounding or truncation of the database numbers published in Boehm's book. For example, project numbers (PN) 4, 8, 14, 16, 20, 24, 31, 33, 34, and 35 had adjustment factor products (Π) more than 1% different than the actual product of the adjustment factors (LANG through RVOL) (Boehm:496). Although the difference seemed small, these differences were well within the resolution of the published numbers. The different results could also be a result of Boehm adjusting the coefficients and exponents based on his personal knowledge and experience.

Data

In order to calibrate the model equations, we attempted to obtain actual historical data from several aircraft support lifecycles. The data collection efforts began in October 1992 by contacting representatives from each of the combat aircraft listed in Chapter I (page 1.12). The initial contacts, from the System Project Offices (SPOs) at Wright-Patterson AFB, lead us to the Air Logistics Centers at Warner-Robins, Ogden, Oklahoma City, and Sacramento. Except for Warner-Robbins, we established good contacts at the ALCs, mailed to them our data forms (see Appendix B), and received a 75% response rate against our targeted platforms. The forms were mailed to ALC engineers involved in software support management for the F-16, the B-52, the F-111, and the B-1. We received no response for the B-1 but received an unsolicited response for the E-3.

Data Collection. Conversations with the ALC contacts indicated that the bulk of the needed data resided in local databases. Despite these promising beginnings, the data received was surprising. Given the slow response time and the sparseness of data, the local databases apparently were not readily accessible or useable and did not contain the needed data. Eleven data points were received, but only four were useful.

Table 4.3 is a summary of the change size/effort data received. The data points are shown in random order and are labeled A - K.

Table 4.3

Actual Data

LABEL	SLOC CHANGED	SLOC AT START OF CYCLE	MANHOURS	DURATION (MONTHS)
A	not available	not available	33,405[1]	13[1]
B	32,300	>500,000	19,500	12
C[2]	0	13,400	2,976	16
D	13,000	not available	17,580	12
E[2]	600	13,000	1,275	11
F	36,626	not available	414,484	29
G[2]	250	10,000	3,000	11
H	not available	not available	27,622[1]	12[1]
I	not available	not available	10,175[1]	8[1]
J[2]	20	41,000	2,325	11
K	31,888	not available	398,105	43[1]

[1] Estimated data not actual.

[2] Usable data points.

Since some participants requested that their data not be traceable to them, the original data sheets were not printed in this thesis.

Only four of the eleven data points contained suitable size and effort information. Unfortunately, four data points were insufficient to validate Hypothesis 1 (functional bottom-up construction of a software cost estimation model) or Hypothesis 2 (functional top-down calibration of a statistically-based software cost estimation model). In most cases, sample sizes of at least 10 or more are necessary to obtain meaningful statistical confidence intervals and to evaluate the statistical measures in Chapter III. Obtaining larger samples required generating a sample database. The last historical data came in May 1993. At that time, we stopped trying to collect additional data to validate our hypotheses and redirected our efforts toward generating sample data to demonstrate the hypotheses.

Data Generation. To generate realistic data for the support model, we invoked the model itself. The model, an enhanced adaptation of Boehm's COCOMO maintenance model, could be considered a mathematical function with multiple inputs (size and attribute factors) and a single output (block change effort). The basic concept was to generate a perfect set of output-input pairs and then add a known variation onto the effort. The resulting fictional sample could then be used to demonstrate both hypotheses.

Starting with the basic concept, we programmed a software module that used the our support equations to generate random data points. The

first step was to generate values simulating the result of a random development product. This was equivalent to generating all the values for the right-hand side of the intermediate COCOMO maintenance equation discussed in Chapter II (equation 2.11). The desired effort-size relationship was the basis for selecting values for a coefficient, a, and an exponent, b. Next, a random number function produced a development product size, *KDSI*, and each of the adjustment attributes, C_{ij}. The prototype calculated three temporal effects, *LE*, *ME*, and *TE*, based on the block change number and then scaled the AEXP, LEXP, STOR, and TIME adjustment attributes. After scaling, the prototype applied three percentages, representing the size of added, modified, and deleted code for a block change, to the *KDSI* size to produce a block change size in units of KDSI. The model then weighted these three sizes according to equation 4.1 to arrive at *EDSI*. Next, dividing *EDSI* by *KDSI* produced *BCT*. At this point, all the values required for the right-hand side of equation 4.5 were calculated, so the prototype calculated the MM, labeled $MM_{estimated}$, for the left-hand side of the equation.

By calculating an effort for each set of inputs associated with a block change, the prototype completed the first half of the basic concept, generating a perfect set of output-input pairs. For the second half of the basic concept, adding a known variation onto the efforts, the prototype applied a random number function. The function accepted the mean and standard deviation of a desired normal distribution and returned an

appropriate random deviate. A mean of zero for all deviate calculations insured an even spread of positive and negative deviate values. Our first inclination was to add the resulting deviate directly to the estimated block change effort, but we realized that adding deviates to effort in the exponential domain was wrong. Since calibration is based upon a linear regression in the logarithmic domain, the perfect input-output pairs had to be converted to the logarithmic domain (as shown by equation 4.10) before adding the deviates.

$$\ln(MM_{estimated}) = \ln(BCT) + \ln(a) + b * \ln(KDSI) + \Sigma \ln(\Pi) \qquad \text{Eq. 4.10}$$

After generating a normally-distributed random deviate, the prototype added the deviate to the converted block change effort. In a sense, adding the random deviate is like adding noise to a perfect block change effort in order to introduce some realism. The resulting effort was labeled $MM_{simulated}$ as shown in equation 4.11.

$$\ln(MM_{simulated}) = \ln(MM_{estimated}) + noise \qquad \text{Eq 4.11}$$

Finally, the prototype model transformed the randomized effort back to the exponential domain by taking the inverse of the natural logarithm (e^x). The generated data could now be treated as historical data for the calibration and evaluation of the model.

Demonstration

Using the data generation methodology, the prototype model created data to demonstrate both hypotheses. By programming the methodology into software using the Visual Basic language, the model created the data sets printed in Appendix F. The data requirements of the hypotheses were different and necessitated two different databases. The simplest database supported the second hypothesis, functional calibration. The more complex database supported the first hypothesis, construction of a bottom-up cost model. The following description for each database generation begins with the Hypothesis 2 (functional calibration) database generation.

Hypothesis 2. The functional calibration hypothesis was demonstrated using families of data points with each family member generated from the same coefficient/exponent combination. Using the Create Database function in the software, the prototype generated five families of 20 data points each. Each family had a different coefficient/exponent combination. By selecting a family or group of families, we were able to simulate how functional calibration might improve software effort estimation and demonstrate the model comparison methodology described in Chapter III.

The prototype constructed the composite database of block change families with inputs shown in Table 4.4.

Table 4.4

Input Parameters for Composite Database

Parameter		Value
Number of Block Changes per Category		20
Development KDSI Size		100
Adjustment Factor Standard Deviation		0.17
Code Affected per Block Change	Percent Added	6
	Percent Modified	10
	Percent Deleted	2
	Percent Std Dev	0.3
Communication Identification	Coefficient	2.40
	Exponent	1.05
	Effort Std Dev	0.4
Navigation Sensors	Coefficient	2.70
	Exponent	1.16
	Effort Std Dev	0.4
Core Avionics	Coefficient	2.70
	Exponent	1.16
	Effort Std Dev	0.4
Electronic Combat	Coefficient	3.10
	Exponent	1.28
	Effort Std Dev	3
Offensive Sensors	Coefficient	3.20
	Exponent	1.36
	Effort Std Dev	0.4

Note that Table 4.4 navigation and core avionics categories were set to the same base equation (coefficient and exponent) and that the standard deviation for the electronic combat category was much higher than the

other categories. After accepting data according to the inputs in Table 4.4, the prototype produced the data shown in Figure 4.3.

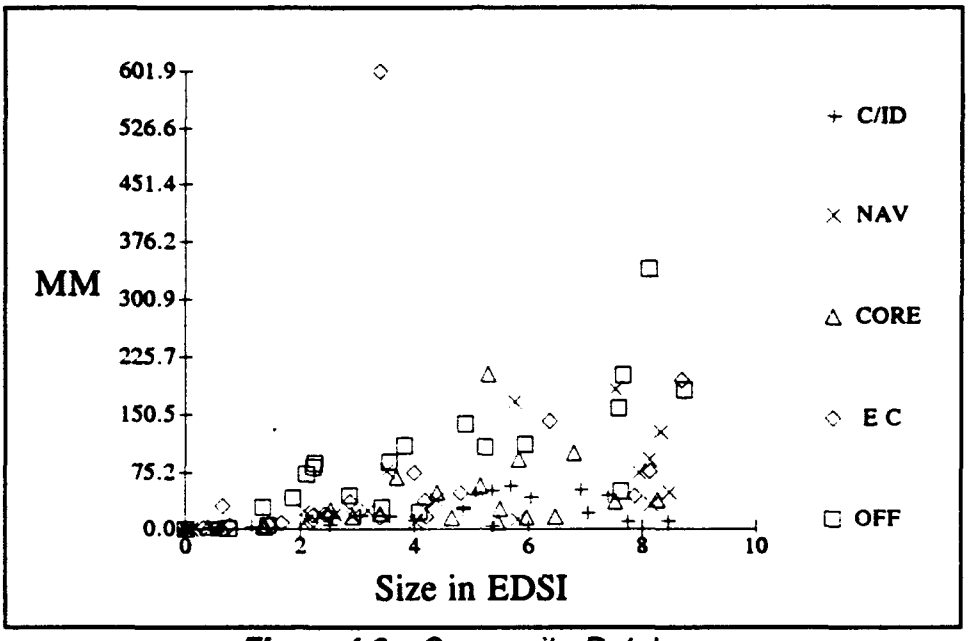


Figure 4.3 - Composite Database

The data in Figure 4.3 had many sources of variation. The first source of variation came from the attribute adjustment factors, which accounted for a large portion of the total variation. The second source of variation came from random proportions of the block change code additions, modifications, and deletions. The last source of variation came from the effort standard deviation assigned to each category. The lone data point from the electronic combat (EC) category was the result of the high variation assigned to this category.

When the variation from the attribute adjustment factors was removed, the prototype produced a normalized database. With the product

of the adjustment factors (∏) divided out (normalizing), the underlying relations of the data were much easier to see. Figure 4.4 shows the data from Figure 4.3 after normalization.

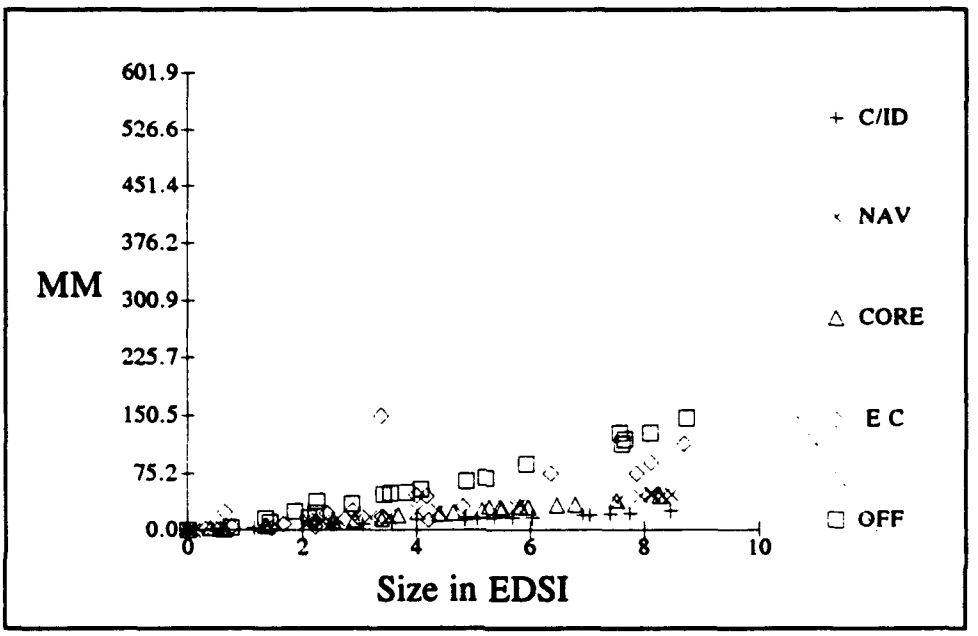


Figure 4.4 - Normalized Composite Database

Notice that many of the data points if Figure 4.4 have aligned along imaginary lines. We exploited this tendency by separating out these points and recalibrating to only these points. Also, many of the data points have lower effort values because they have a total product adjustment (∏) greater than 1.0. To compensate for the change in scale, the prototype enlarged the data scale by reducing the maximum plot magnitude of the y-axis to 150 resulting in Figure 4.5.

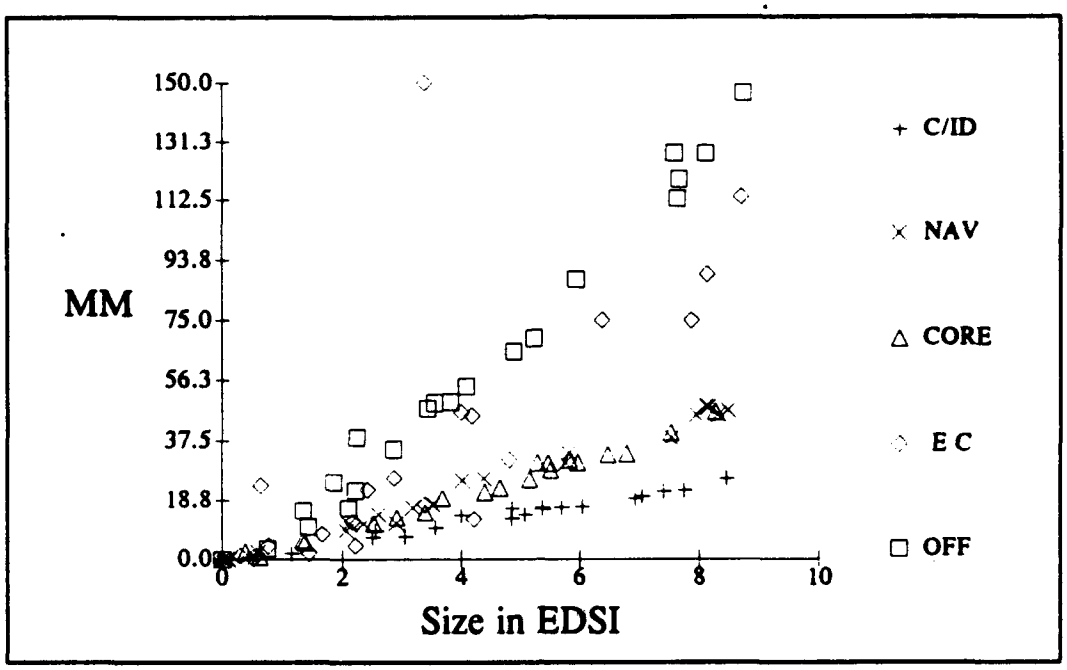


Figure 4.5 - Zoomed and Normalized Composite Database

Figure 4.5 shows the grouping or family relationship more clearly than Figure 4.4 does. The separate families of categories reflect the stratification of the data.

If the data were separated into the different categories and plotted, estimators could easily spot the multiple stratifications. But what if all the data were assumed to be in the same category? If this were true, estimators might be inclined to see a single exponential relationship instead of multiple ones, especially if the variance (spread) of each category was larger. Without further examination, estimators could calibrate the entire set of data and produce a single coefficient and exponent pair for a single model. Calibration of the entire data set yielded a coefficient of 2.43 and an exponent of 1.23. Although an estimation model with these parameters might not account for each of the separate

data categories, the model would still estimate more accurately than a model that was not calibrated to the data set. Table 4.5 contains the results of a comparison of the REVIC embedded mode model (a = 3.312 & b = 1.20) and a model using the calibrated parameters.

Table 4.5

Statistical Evaluation of Calibration Source

Parameter Source	R^2	RRMS	Avg MRE	Predict Value %	% Improve
REVIC	0.7906	1.176	32.81	12.0	—
Data Calibration	0.8072	1.175	28.54	35.0	68%

The results in Table 4.5 demonstrated that estimating effort from calibrated models could be more accurate than estimating from a model that is uncalibrated. In fact, if the calibration is performed properly and the calibration data adequately represents the estimation environment, a calibrated model should *always* estimate more accurately than the same uncalibrated one. Thus, any model that uses statistical correlation can be improved by calibration.

By dividing the data into functional categories, estimators could obtain more accurate coefficient/exponent pairs. For example, calibrating the communication-identification category yielded a coefficient of 1.37 and an exponent of 1.18 while calibrating the offensive sensor category yielded

a coefficient of 2.97 and exponent of 1.39[1]. These values were not the same as those for the entire data set. Applying each of the evaluation measurements from Chapter III produced the results printed in Table 4.6.

Table 4.6

Statistical Evaluation of Categorical Calibration

Calibration Source		R^2	RRMS	Avg MRE	Predict Value %	% Improve
Entire Database	Database	0.8072	1.175	28.54	35.0	—
Comm ID	Database	0.5236	1.506	26.86	0.0	—
	Category	0.9763	0.096	1.79	80.0	100.0
Nav Sensors	Database	0.9110	0.351	9.58	75.0	—
	Category	0.9148	0.113	2.96	90.0	70.0
Core Avionics	Database	0.9310	0.257	7.25	70.0	—
	Category	0.9512	0.15	2.74	90.0	80.0
Elect Combat	Database	0.6790	1.748	44.89	25.0	—
	Category	0.7109	1.630	37.63	50.0	60.0
Offensive Sensors	Database	0.6447	0.740	54.10	5.0	—
	Category	0.9785	0.137	9.06	75.0	95.0

Table 4.6 revealed the potential improvement in all of the measurements by calibrating to a functional data set. Categories at the edges of the data set, such as communication-identification and offensive sensors, did show a pronounced improvement while other categories in the central area of the whole data set did not show as much improvement.

[1] These values are not the same as the input parameters because of the effort variability introduced when generating the data points.

After separating the categories, graphing the families, and evaluating the statistics, it was an easy chore to determine where the improvements were. But what about the areas where there were no improvements? Data collection data for analysis required much effort, so avoiding data collection for any area should save effort. According to the graph of the data (Figure 4.5), the navigation and core avionics functions seemed to overlap closely. Comparing the navigation and core avionics calibration results with other categories in Table 4.7 provided some additional clues.

Table 4.7

Category Calibration Results

Category	Coefficient	Exponent
Comm / ID	1.3689	1.1845
Nav Sensors	2.4279	1.1918
Core Avionics	2.3197	1.1923
Elect Combat	1.7652	1.3969
Off Sensors	2.9665	1.3876
Entire Database	2.4291	1.2278

Table 4.7 confirmed that the coefficients and exponents of the navigation and core avionics categories were close to each other. Therefore, it made sense not to collect data for these two categories separately but to combine them into one group with a single calibration. Calibrating these two categories together as one category produced a coefficient of 2.3925 and an exponent of 1.1898. Neither of these new values were significantly

different from the previous values, so treating both categories as one would not greatly affect the accuracy of any predictions for block changes in these categories. Of course, this prediction assumed that the future behavior of these categories would follow the past performance. This may or may not be true, but an informed guess should be better than an ignorant one.

Another category which would probably benefit little from special calibration attention was electronic combat. Because the variability of this category was high, the category was naturally difficult to estimate. Collecting extra data to enable estimators to calibrate a new parameter pair would probably be less helpful than collecting extra data on a more promising category such as communication-identification. The calibration from the entire data set predicted almost as well as the functional calibration for the electronic combat category.

For the second hypothesis of functional calibration, we have demonstrated the main benefit of producing new prediction relationships based upon a selected subset of data. The subset could be an entire group of data such as avionics or subset families of a group. In either case, the main benefit would be improved prediction accuracy. The generated database separated a fictional database into five functional categories. The communication-identification and offensive sensors calibrations offered the most promising improvements and warranted their own calibration curves. On the other hand, the navigation, core avionics, and electronic

combat categories demonstrated less potential for improved accuracy and might be better treated as a single group.

Hypothesis 1. Functional calibration for the second hypothesis was not the only way to demonstrate prediction accuracy improvements. By generating data that fit the first hypothesis, construction of a bottom-up cost model, we demonstrated another possible improvement. Creating a database for the first hypothesis required a series of block changes for a lifecycle. Each lifecycle had to mirror the temporal changes suggested within the literature review. The concept of generating a series of data points was similar to the concept used to create a single data point. The prototype determined the effort for a particular block change size and then added some random deviation. The wrinkle this time was to change the support effort to match the bath-tub curve espoused in the literature. Recall from Chapter II the four temporal effects which impacted the size/effort relationship and made the bath-tub curve. The prototype modeled the memory effect, throughput effect, and the learning effect but discarded the entropy effect because there were no quantifiable relationships for entropy. The bath-tub shaped lifecycle came entirely from the first three temporal effects.

The prototype constructed a series database with the inputs shown in Table 4.8.

Table 4.8

Input Parameters for Series Database

Parameter		Value
Number of Block Changes		20
Development Size in KDSI		100
Development Memory Utilization Percentage		50.0
Development Throughput Utilization Percentage		50.0
Effort Standard Deviation		2.0
Adjustment Factor Standard Deviation		0.17
Prediction Parameters	Coefficient	2.80
	Exponent	1.20
Percentages of Total Size for EDSI calculation	% Added	5.0
	% Modified	10.0
	% Deleted	2.0
	% Standard Deviation	0.3

We selected the input values in order to generate a data set that could demonstrate the effects discussed in the literature review. The resulting data can be found in Appendix F and in Figure 4.6.

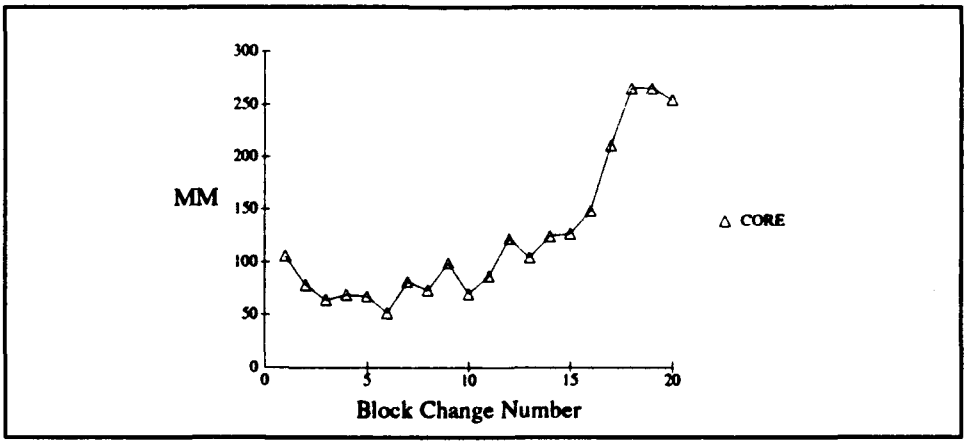


Figure 4.6 - Series Block Change Database

The most noteworthy characteristic of this data was the initial decrease in effort during the first six block changes followed by an increase in effort towards the later block changes. By normalizing the data to remove the variation induced by changing adjustment factors, the prototype more clearly displayed the temporal effects in Figure 4.7.

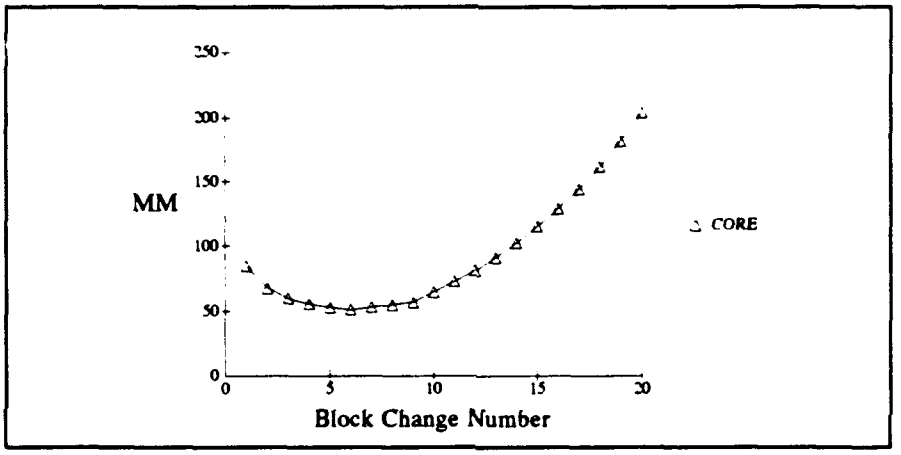


Figure 4.7 - Normalized Series Block Change Database

Figure 4.7 demonstrated what happened to avionics support effort as the learning effect died out and was overtaken by the effects of depleting the remaining throughput and memory. The effort for the last block change was more than double the effort from the first block change. If the temporal effects were accurately modeled, then it was easy to understand why models which don't include these effects could be less accurate. A model that simply assumed a constant level of effort could not account for learning or any effect of growth.

Although the data in Figure 4.6 does not deplete all of the remaining memory or throughput, a data set which did approach the 100% limits demonstrated the value of controlling growth. By adjusting the percent added code from 5% to 6% and holding the other inputs constant, the prototype demonstrated the growth effects as graphed in Figure 4.7.

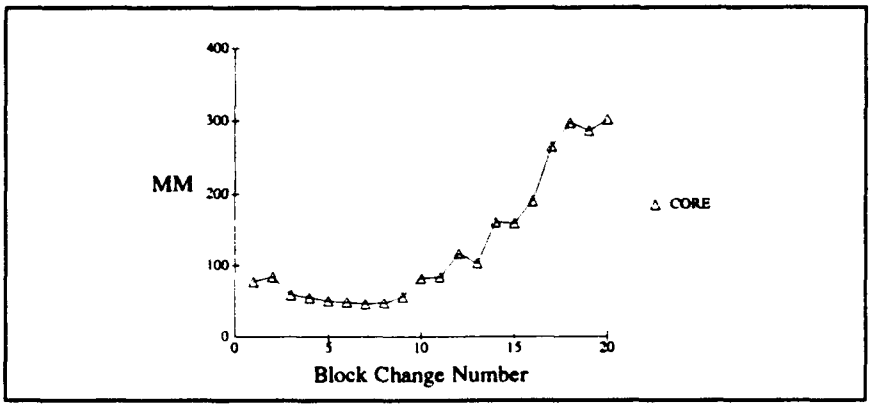


Figure 4.8 - Support Data at 6% Added Code

The effort leveling during the last three block changes demonstrated the effect of memory and throughput utilization approaching 100%. Operating in this area is very costly. Figure 4.9 displays the memory utilization for separate examination.

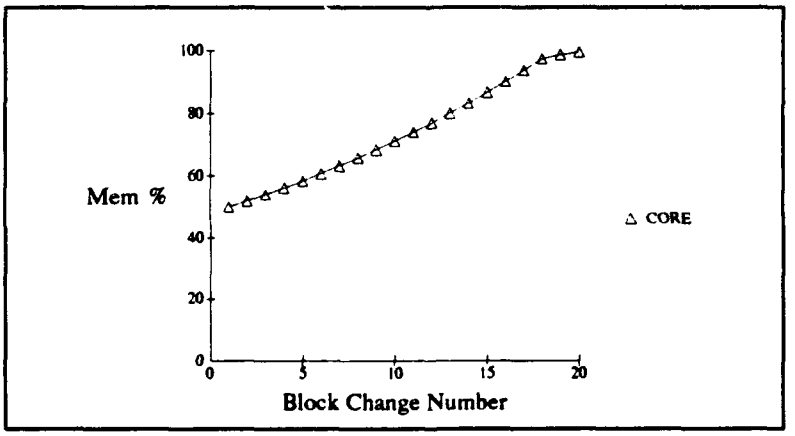


Figure 4.9 - Memory Utilization Growth

The memory utilization grew steadily until reaching about 95% where it began to taper. The throughput utilization exhibited a similar growth pattern. Although the consumption of these two resources significantly altered the later portions of this support lifecycle, the high effort was partially offset by the learning which occurred during the early portion of the lifecycle. Figure 4.10 displayed the values for the language experience (LEXP) adjustment factors as altered by the learning effect.

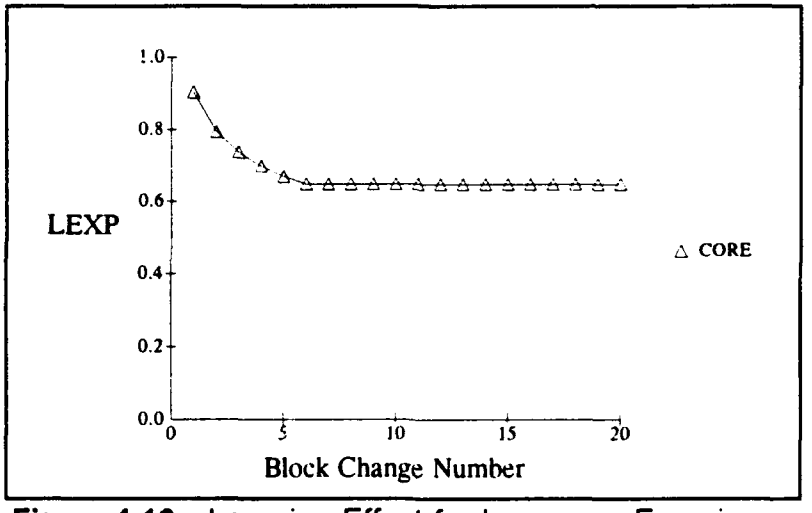


Figure 4.10 - Learning Effect for Language Experience

The learning effect was manifested only during the first six block changes, and after that point, it did not help reduce the require effort. Since all three temporal effects influenced effort by altering adjustment factors, it was worthwhile to examine the total contribution of the temporal effects by graphing the adjustment factor products (Π) for successive block changes. The products are graphed in Figure 4.11.

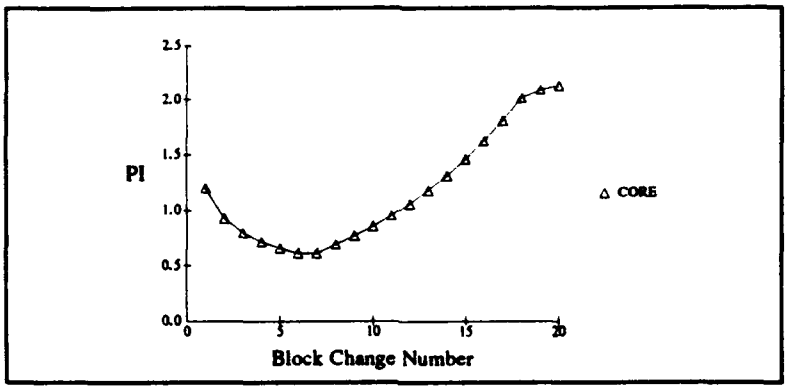


Figure 4.11 - Block Change Adjustment Products

The shape of Figure 4.11 was the same shape of a graph of normalized efforts. This match was not surprising since the change in support effort came entirely through the change in adjustment factors.

After reviewing the results of the software, we concluded the prototype model was an excellent tool to demonstrate the hypotheses. Although actual data would have enabled the software to validate or disprove the relationships, the prototype demonstrated a way and provided a tool for future researchers to make these conclusions.

REVIC/Prototype Model Comparison

Despite the utility of the prototype in demonstrating the hypotheses, we also compared the order of magnitudes of the effort predictions from the model against well-established models. Since the prototype and REVIC models are both adaptations of the COCOMO model, it made sense to compare the outputs from the REVIC model, the COCOMO model and the prototype model. The comparison expanded to include the SASET and

SEER models because those models were readily available. The comparison was simplified by constraining all the adjustment factors in the COCOMO-based models to nominal values (1.0). The inputs of the other models were also nominalized to the extent possible. Nominalizing the model permitted comparing of the results and avoided accounting for model-specific differences. Nominalization also reduced each of the COCOMO-based models to forms similar to the basic COCOMO model presented in Chapter II. The comparison results are shown in Table 4.9.

Table 4.9

Support Model Comparisons

		REVIC	COCOMO	SASET	SEER	Proto	Proto
Inputs	Block Changes	15	15	15	15	15	15
	Devel Size	100 KDSI	100 KDSI	100 KDSI	100 KDSI	100 KDSI	100 KDSI
	Block Change Size	10% ACT	10% ACT	10% ACT	10% ACT	0% add 20% md 0% del	6% add 12% md 2% del
Outputs	Effort						
1	(MM)	124.8	83.19	63.72	393.84	84.9	84.9
2		108.2	83.19	54.12	243.36	65.4	68.6
3		95.7	.	46.2	167.78	56.2	61.8
4		83.2	.	37.56	167.78	50.5	58.1
5		.	.	31.44	.	46.4	56.0
6		.	.	28.8	.	43.3	54.9
7		.	.	26.16	.	43.3	57.5
8		.	.	22.68	.	.	67.9
.		.	.	.	.	.	.
.		.	.	.	.	.	.
13		.	.	8.76	.	.	146.7
14		.	.	6.12	.	.	171.1
15		83.2	83.19	3.48	167.78	43.3	199.6

The comparison results printed in Table 4.9 indicated that the prototype model predicted efforts that were neither the highest nor the lowest of the well-established models. Thus, the prototype generated effort values within an order of magnitude of the other models.

Conclusion

Although we would have preferred using actual data, we accomplished all of the objectives using generated data. Demonstration of Hypothesis 1 began by modeling the current block change process using object oriented design techniques. The block change process model and the literature review provided the foundations for creating a software computer model. The prototype demonstrated the improved accuracy of a bath-tub shaped series of block changes and an order of magnitude accuracy compared to COCOMO, REVIC, SASET, and SEER. These actions satisfied the first three objectives for Hypothesis 1 as stated in Chapter I.

We also achieved the first objective for Hypothesis 2 as stated in Chapter I. The prototype demonstrated the improved accuracy of functionally calibrating software support estimation models. The remaining objectives for both hypotheses are recommendations that are covered in the next chapter.

Lack of data was an unforeseen obstacle that we overcame with the prototype by generating fictional data. Actual data existed but not in a

readily accessible form. The lack of data forced a deviation from the original concept of validating the model to demonstrating the prototype.

V. Conclusions & Recommendations

Chapter Overview

In this chapter, we present conclusions that were supported by our research. Also presented are recommendations for improving software support estimation. The conclusions and recommendations are grouped according to the topics of data: Hypothesis 1 (functional model development), Hypothesis 2 (functional calibration), and the model comparison methodology. The chapter concludes by identifying other topics for additional research.

Data

Conclusions. We were unable to find sufficient data through direct collection and through literature research to confirm or refute the hypotheses. However, our efforts supported two conclusions. The first conclusion was that the efficiency of an SSA will not remain constant over the PDSS lifecycle for aircraft avionics software. The second conclusion was that a software support database needs to be constructed to help study long-term SSA efficiency. Each conclusion is discussed below.

The nature of the software support process coupled with the constraints imposed by a target computer suggested that a bath-tub shaped curve existed which was not captured in REVIC, SEER, or SASET maintenance models. However, the literature confirmed the existence of

this curve. There were four factors that affected the shape of this curve. The first was a learning effect acquired through exposure to the software. The second was simple size growth during the PDSS cycle. The third was the loss of available memory and throughput as size increased. The fourth was design entropy caused by repeated changes to the software. Further research into each of these factors could lead to understanding the temporal effects and to identifying any interdependencies.

The second conclusion was the need for a support database to quantify the four temporal effects. A database is a prerequisite to support model development and PDSS estimation improvements. The data uncovered during the research was insufficient to validate the hypotheses and forced the generation of sample data. Until sufficient data is collected for analysis, no further progress can be made on either hypothesis, nor can an evaluation of the existing models be accomplished. The specific attributes of the database are discussed in the next section.

Recommendations. Building an adequate support database will be a challenge. The database must have time series information on a sufficient number of software projects to allow the revision and calibration of present models to a support environment. The database must also contain high quality information and requires a well-defined data collection plan that must be strictly followed. Each database element must be precisely defined to reduce confusion during collection and interpretation. Besides these general characteristics, we made specific recommendations about

collecting size and adjustment factors. Size, the most important parameter, will be discussed first.

Good size data is the cornerstone for a good PDSS database. As mentioned in Chapter II, all the other factors simply adjusted the effort estimate obtained from size. Support size was similar to development size, but there were some key differences. Both support and development size required a standard definition. It is not sufficient to simply ask for size in SLOC, KDSI, function points, or any other size units without a standard definition of the metric. Implied in this statement is a software language dependency; the definition of size is likely to be highly language dependent. However, the goal is to define the metric so that two different estimators will generate the same size count for a given piece of software. One solution would be to embed the sizing rules into a software tool and to use it on each support tape entered into a PDSS database. This would provide an accurate starting size for each block change. Now given the starting size of a baseline, estimators could calculate the effort per EDSI observed in the block change. At this point, the recommendations diverge from sizing techniques common to development.

Support software size has an additional measurement, the size of the changed code, that development size does not have. Software support estimates should derive effort from the size of the changed code as well as the total code size. A consistent way to measure the change size would be to compare a baseline tape with a tape produced after a block change

cycle. All of the code that was added, deleted, or modified must be counted. Again, the counting metrics must be defined clearly so they can be applied uniformly. Of all the possible size measures, SLOC (or KDSI) is a measure that is usually familiar to software engineers since they usually work with lines of code (or source instructions). Therefore, we recommend that a set of language-specific definitions for added, deleted, and modified SLOC be created and codified into a standard software tool for measuring the size of baseline code and for measuring changes to the code. Only size measured by the standard tool should be allowed into the PDSS database.

Along with size data, adjustment factors need to be gathered to account for other variables present in the block change. Until a better support model is available, we do not recommend a standard set of adjustment factors for data collection. In the mean time, collecting all of the support adjustment factors from the Air Force recommended models (REVIC, SASET, SEER, and PRICE-S) should suffice. We made no conclusion which models or attributes to drop from the PDSS database. However, three critical areas should be included in any database. As with size, each of these areas needs a defined metric to ensure data consistency. The first critical area is the experience of the analysis, design, and coding team. The team size and average experience in months at the start of the block change cycle should be part of the PDSS database. This information is needed to study the learning effect.

The second critical area is the remaining memory and throughput. Although measuring these attributes can be complex and highly machine dependent, the important characteristic to capture is the designer's perception of remaining memory and throughput. It is equally vital to record any hardware changes that increase available memory and throughput. Expressing the available memory and throughput as a percentage is reasonable given the complexity of the measurements.

Even more complex to measure than remaining memory and throughput is the third critical area, design entropy. Although all the factors that contribute to design entropy unknown, one cure for design entropy is to re-engineer the software (Sittenauer:7-10). Another less drastic cure is to restructure the software (Corbi:294-306). Unfortunately, no single metric existed that captured all of the software attributes that restructuring and re-engineering restore. But there are many metrics that can capture a subset of those attributes (Oman:337-344, Fenton:150-259). The question is which metrics capture design entropy with reasonable data collection. We recommend that the PDSS database retain the baseline code and documentation for each block change. We also recommend further investigation into the nature and measurement of design entropy to define a set of metrics for inclusion in a PDSS database.

A PDSS database is not the only method of researching maintenance productivity over time. Further literature research into the learning curve, memory/throughput effects, and design entropy effects may prove

fruitful. There may be other factors not identified in this thesis that contribute to the bath-tub temporal effects.

Hypothesis 1: Functional Construction of a Software Estimation Model

Conclusions. Describing the block change process using software design tools provided excellent results. The object model labeled the agents involved in the PDSS environment while the functional model described the subprocesses within the block change process. The graphic nature of our object model allowed quick assignment of the COCOMO effort adjustment factors to specific agents in the PDSS environment, while the graphic nature of the functional model allowed rapid identification and resolution of differences between block change process descriptions. The result was a product that was traceable to MIL-HDBK-347 and to previous work done by Sacramento ALC. The next task was to functionally construct a software estimation model based on the block change process described by the process model.

The concept of a functionally constructed software estimation model, given a functional description of the process, is to estimate each function (subprocess) individually and then sum the total. Once the block change process was correctly depicted, the subprocesses best represented by the COCOMO maintenance model were included in the prototype software. The prototype included only the "develop changes" subprocess because there was insufficient data to include other subprocesses.

Estimators in the field should included their own selection of subprocesses based on the data available for calibration and for other estimation methods (analogy or bottom-up). Having identified what part of the block change process the prototype would address, we next manipulated the COCOMO adjustment factors to mirror the bath-tub curve supported by literature.

The software prototype successfully created a bath-tub curve over a series of block changes. The fidelity and accuracy of the prototype compared to an actual series of block changes was unknown. With no usable data to test against the prototype, we could only assert that the prototype represented the temporal effects of software support better than the ACT based models of REVIC, SASET, and SEER did.

Recommendations. We have two recommendations regarding the prototype model. The first is to validate and calibrate it with quality data as soon as possible. This is necessary to both confirm the existence of the bath-tub curve. Once the prototype is validated and calibrated, the second recommendation is to prompt model designers to update the existing support process models to incorporate a bath-tub curve.

Hypothesis 2: Functional Calibration to Improve Model Accuracy

Conclusions. Calibration of support (or development) estimation models is essential. The closer the calibration data matches the expected system, the closer the estimate should be. The importance of proper

calibration is under emphasized. Unfortunately, calibration for support models is not as simple as calibration for development models.

Support model calibration models is feasible and can take two forms, horizontal and vertical. Each form takes a different temporal slice of the PDSS lifecycle. Horizontal calibration captures *all* of the long-term, stable characteristics of a PDSS lifecycle. This is critical for estimators to understand and is the key to identifying unique adjustment factors for an SSA. Long-term calibration may alleviate the need for some of the adjustment factors. Models do not need variables that account for attributes which remain constant over the long-term. Vertical calibration is similar to developmental calibration. Each uses the i^{th} development cycle from a number of systems to capture those processes common across those systems for that particular cycle.

Recommendations. The first recommendation to include ease of calibration as a criterion to judge estimation models. A model that cannot be easily calibrated loses much of its utility. If a model can't be easily calibrated, then the estimator must know what data was used to calibrate the model originally and realize that estimating effort outside of this data domain is likely to be inaccurate.

The second recommendation is to use models that can be calibrated in a repeatable manner. One test of model calibration is to use the calibration procedure with the original data to see if it is repeatable. If

the recalibrated model doesn't match the original, then there is a serious problem that needs resolution before using the model.

The third recommendation is to avoid using a model without understanding how it was calibrated and with what data. This information is vital to understanding what bias might exist in an estimate. This is a serious problem with proprietary models with no simple solutions. However, a quality Air Force database can help solve this problem. Given a quality database, we recommend calibration of a model to that database prior to estimating, especially if the model's calibration database is unknown to the estimator.

Our fourth recommendation is to train estimators. Estimators need to be trained how to calibrate the models, especially if the models use a complex calibration procedure. Again, a readily available database is needed before training in model or else calibration will be ineffective. Additional training on the nature of software, software metrics, and the processes used to create software should help estimators create better estimates. If nothing else, the training will help software engineers and estimators communicate better to improve estimation.

The final recommendation centers is to exploit the PDSS database when it becomes available. Long-term data from a PDSS database will highlight factors that are best captured under calibration. Using analysis of variance (ANOVA) techniques on all the factors should reveal those adjustment factors that are sufficiently stable over time to be firmly

captured in calibration. This calibration should reduce the number of variable factors that need to be maintained in the database.

Future Research Topics

The single most important research topic is block change size estimation. Size is the driver of all the Air Force recommended models. Further research could determine the proper weighting for changed lines, added lines, and deleted lines to produce a support size with the highest correlation to support effort. Researchers also need to investigate better methods of size estimation in preparation for a block change since block change efforts do not come from measured EDSI inputs but from estimate EDSI inputs. Better effort estimates require adequate models and quality size estimates.

Summary

While software support estimation is a fertile field for future research, the key to future research is in creating a support database. Once a large enough database is gathered, the current generation of software support models can be calibrated and validated to specific environments. Estimators can make further improvements to the current estimation models for avionics software support by incorporating temporal effects such as learning, size growth, memory/throughput growth, and design entropy.

In this thesis, we reviewed the current literature dealing with software cost estimation models and the software support environment. We also presented methodologies to document the support process and to compare software cost estimation models. Furthermore, we created a software prototype that embodied the PDSS lifecycle as described in the literature. The prototype generated data to demonstrate how current software cost estimation models could be improved from two functional points of view, functional bottom-up model design and functional top-down model calibration. We demonstrated that both techniques can improve cost estimation accuracy and provided recommendations for data collection, model improvement, and future research.

Appendix A

Object Oriented Model Notation Summary

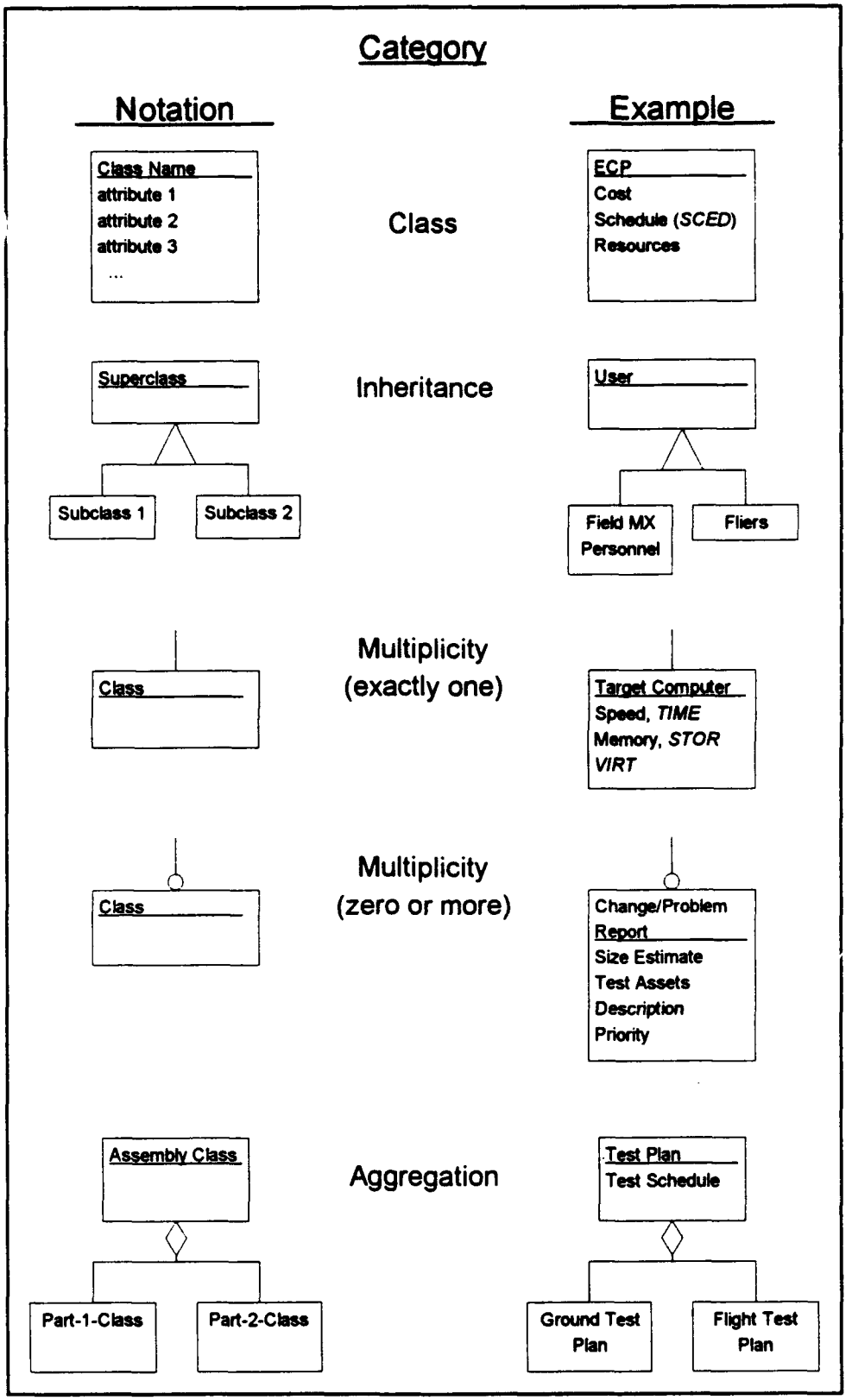


Figure A.1 - Object Model Notation

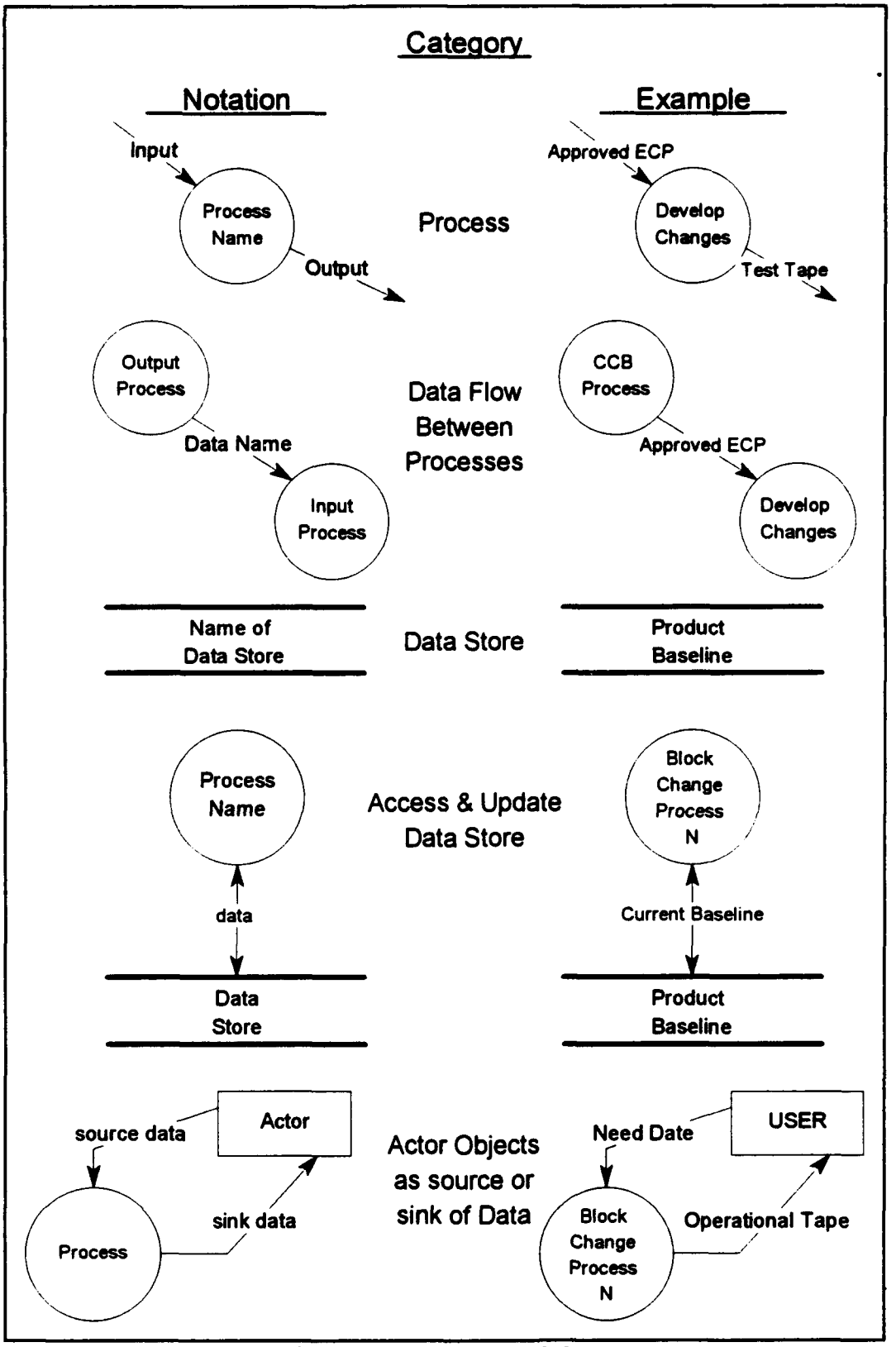


Figure A.2 - Functional Model Notation

Appendix B

Data Collection Forms

Data Collection Form

Name: _____ Organization: _____

a) Block Change Identification Number: _____

b) Configuration Element:

CSCI	☐	Check box
CSC	☐	Describe Others below
CSU	☐	
Other	☐	_____

c) Software Language — Percent of Total Code

1) _____ _____ %
2) _____ _____ %
3) _____ _____ %

d) System Lines of Code (SLOC)

Answer for this Block change only.

Start Total SLOC	
New SLOC	
Revised SLOC	
Deleted SLOC	
End Total SLOC	

e) Total Manhours for this block change: _____

f) Schedule

Start Date: _____ End Date: _____

Milestone	Date

g) Software Support Environment (describe below or attach description)

TOOLS:
DOCUMENTATION:

h) Team Experience (average in years) with:

The software	
The documentation	
The environment	
The language	

i) Description of the software functions (describe below or attach description)

Description of changes (describe below or attach description)

j) Target Computer Description (describe below or attach description)

Available Memeory
(at start of change)

Available Throughput
(at start of change)

k) Estimation Methods
Models

REVIC/COCOMO		Check box
SASET		
SEER		Describe Others below
PRICE S/PRICE SM		
Other		

Estimation Process Description (describe below or attach description)

Other Information/General Comments

Appendix C

Prototype Source Code

Source Code for Prototype in Visual Basic Language.

Code for Global.bas

```
'Enumerated Database type
Global Const UnknownDB = 0
Global Const BoehmDB = 1
Global Const SampleDB = 2
Global Const ThesisDB = 3
Global Const CompThesisDB = 4

'Set up Enumeraated Values for YData
'Only the thesis graph needs to use this
Global YPick As Integer
Global Const Executable_Time = 1
Global Const Time_Util = 2
Global Const Storage = 3
Global Const Mem_Util = 4
Global Const Pi_Mult = 5
Global Const Analyst_Experience = 6
Global Const Lang_Experience = 7
Global Const Actual_KDSI = 8
Global Const Equivalent_DSI = 9
Global Const Actual_Effort = 10

Global Const FileGraph = 0

'Global Variables for Graph Axis Limits and Normalization Flag
'Should only be used by forms involved with graphing
'(I couldm't figure another way to pass these parameters.)
Global YMax As Single
Global YMin As Single
Global XMax As Single
Global XMin As Single
Global NormState As Integer
Global Const Norm_On = 1
Global Const Norm_Off = 0

'Provide Global Variable to hold current open database filename
'and Handle.  Permits different menu items to use same database.
Global DBFileSpec As String
Global DBFileName As String
Global DBHandle As Integer
Global more As Integer          'flag stating if more records exist
Global DBType As Integer        'Enumerated Database Type
Global ErrMsg As String         'string for error mesaages

'Define Boehm's record structure from database
Type BoehmType
    Num         As Variant      'variant type permits testing of null values
    Type        As Variant
    YEAR        As Variant
    LANGUAGE    As Variant
    RELY        As Variant
    DBSIZE      As Variant
    CPLX        As Variant
    EXTIME      As Variant
    STOR        As Variant
    VIRT        As Variant
    TURN        As Variant
    ACAP        As Variant
    AEXP        As Variant
    PCAP        As Variant
    VEXP        As Variant
    LEXP        As Variant
    MODP        As Variant
    TOOL        As Variant
    SCED        As Variant
    RVOL        As Variant
    Pi          As Variant
    MODE        As Variant
    TOTKDSI     As Variant
```

```
    ADJKDSI     As Variant
    NOMEFFORT   As Variant
    ESTEFFORT   As Variant
    ACTEFFORT   As Variant
End Type
Global BoehmRec As BoehmType
Global BoehmHandle As Integer
Global Const BoehmRDS = "(PROJ_NUM(V) PROJ_TYPE(V) YEAR(V) LANGUAGE(V) RELY(V) DBSIZE(V) CPLX(V)
EXTIME(V) STOR(V) VIRT(V) TURN(V) ACAP(V) AEXP(V) PCAP(V) VEXP(V) LEXP(V) MODP(V) TOOL(V) SCED(V)
RVOL(V) PI(V) MODE(V) TOTKDSI(V) ADJKDSI(V) NOMEFFORT(V) ESTEFFORT(V) ACTEFFORT(V))"

'Define Short version of Beohm's type for creating sample databases
Type SampleType
    Num         As Variant      'variant type permits testing of null values
    Type        As Variant
    RELY        As Variant
    DBSIZE      As Variant
    CPLX        As Variant
    EXTIME      As Variant
    STOR        As Variant
    VIRT        As Variant
    TURN        As Variant
    ACAP        As Variant
    AEXP        As Variant
    PCAP        As Variant
    VEXP        As Variant
    LEXP        As Variant
    MODP        As Variant
    TOOL        As Variant
    SCED        As Variant
    RVOL        As Variant
    Pi          As Variant
    ACTKDSI     As Variant
    ACTEFFORT   As Variant
End Type
Global SampleRec As SampleType
Global SampleHandle As Integer
Global Const SampleRDS = "(PROJ_NUM(V) PROJ_TYPE(V) RELY(V) DBSIZE(V) CPLX(V) EXTIME(V) STOR(V)
VIRT(V) TURN(V) ACAP(V) AEXP(V) PCAP(V) VEXP(V) LEXP(V) MODP(V) TOOL(V) SCED(V) RVOL(V) PI(V)
ACTKDSI(V) ACTEFFORT(V))"

'Define Thesis record Structure
Type ThesisType
    BCNUM       As Variant      'variant type permits testing of null values
    CATEGORY    As Variant
    RELY        As Variant
    DBSIZE      As Variant
    CPLX        As Variant
    TIMEUTIL    As Variant
    EXTIME      As Variant
    MEMUTIL     As Variant
    STOR        As Variant
    VIRT        As Variant
    TURN        As Variant
    ACAP        As Variant
    AEXP        As Variant
    PCAP        As Variant
    VEXP        As Variant
    LEXP        As Variant
    MODP        As Variant
    TOOL        As Variant
    SCED        As Variant
    RVOL        As Variant
    Pi          As Variant
    ENTROPY     As Variant
    ACTKDSI     As Variant
    ADDKDSI     As Variant
    MODKDSI     As Variant
    DELKDSI     As Variant
    EDSI        As Variant
    ACTEFFORT   As Variant
End Type
Global ThesisRec As ThesisType
Global ThesisHandle As Integer
```

```
Global Const ThesisRDS = "(BCNUM(V) CATEGORY(V) RELY(V) DBSIZE(V) CPLX(V) TIMEUTIL(V) EXTIME(V)
MEMUTIL(V) STOR(V) VIRT(V) TURN(V) ACAP(V) AEXP(V) PCAP(V) VEXP(V) LEXP(V) MODP(V) TOOL(V) SCED(V)
RVOL(V) PI(V) ENTROPY(V) ACTKDSI(V) ADDKDSI(V) MODKDSI(V) DELKDSI(V) EDSI(V) ACTEFFORT(V))"

'Define BadThesis record Structure
Type OldThesisType
    BCNUM       As Variant      'variant type permits testing of null values
    CATEGORY    As Variant
    RELY        As Variant
    DBSIZE      As Variant
    CPLX        As Variant
    TIMEUTIL    As Variant
    EXTIME      As Variant
    MEMUTIL     As Variant
    STOR        As Variant
    VIRT        As Variant
    TURN        As Variant
    ACAP        As Variant
    AEXP        As Variant
    PCAP        As Variant
    VEXP        As Variant
    LEXP        As Variant
    MODP        As Variant
    TOOL        As Variant
    SCED        As Variant
    RVOL        As Variant
    Pi          As Variant
    ENTROPY     As Variant
    ACTKDSI     As Variant
    ADDKDSI     As Variant
    MODKDSI     As Variant
    DELKDSI     As Variant
    ESDI        As Variant
    ACTEFFORT   As Variant
End Type
Global OldThesisRec As OldThesisType
Global OldThesisHandle As Integer
Global Const OldThesisRDS = "(BCNUM(V) CATEGORY(V) RELY(V) DBSIZE(V) CPLX(V) TIMEUTIL(V) EXTIME(V)
MEMUTIL(V) STOR(V) VIRT(V) TURN(V) ACAP(V) AEXP(V) PCAP(V) VEXP(V) LEXP(V) MODP(V) TOOL(V) SCED(V)
RVOL(V) PI(V) ENTROPY(V) ACTKDSI(V) ADDKDSI(V) MODKDSI(V) DELKDSI(V) ESDI(V) ACTEFFORT(V))"

''''''''''''''''''''''''''''''
' Visual Basic global constant file. This file can be loaded
' into a code module.
'
' Some constants are commented out because they have
' duplicates (e.g., NONE appears several places).
'
' If you are updating a Visual Basic 1.0 program to run in
' Visual Basic 2.0, you should replace your global constants
' with the constants in this file.  Note that True and False
' are now built into Visual Basic so are no longer defined in
' this file.
'
''''''''''''''''''''''''''''''

' General

' Clipboard formats
Global Const CF_LINK = &HBF00
Global Const CF_TEXT = 1
Global Const CF_BITMAP = 2
Global Const CF_METAFILE = 3
Global Const CF_DIB = 8
Global Const CF_PALETTE = 9

' DragOver
Global Const ENTER = 0
Global Const LEAVE = 1
Global Const OVER = 2

' Drag (controls)
Global Const CANCEL = 0
Global Const BEGIN_DRAG = 1
```

```
Global Const END_DRAG = 2

' Show parameters
Global Const MODAL = 1
Global Const MODELESS = 0

' Arrange Method
' for MDI Forms
Global Const CASCADE = 0
Global Const TILE_HORIZONTAL = 1
Global Const TILE_VERTICAL = 2
Global Const ARRANGE_ICONS = 3

'ZOrder Method
Global Const BRINGTOFRONT = 0
Global Const SENDTOBACK = 1

' Key Codes
Global Const KEY_LBUTTON = &H1
Global Const KEY_RBUTTON = &H2
Global Const KEY_CANCEL = &H3
Global Const KEY_MBUTTON = &H4     ' NOT contiguous with L & RBUTTON
Global Const KEY_BACK = &H8
Global Const KEY_TAB = &H9
Global Const KEY_CLEAR = &HC
Global Const KEY_RETURN = &HD
Global Const KEY_SHIFT = &H10
Global Const KEY_CONTROL = &H11
Global Const KEY_MENU = &H12
Global Const KEY_PAUSE = &H13
Global Const KEY_CAPITAL = &H14
Global Const KEY_ESCAPE = &H1B
Global Const KEY_SPACE = &H20
Global Const KEY_PRIOR = &H21
Global Const KEY_NEXT = &H22
Global Const KEY_END = &H23
Global Const KEY_HOME = &H24
Global Const KEY_LEFT = &H25
Global Const KEY_UP = &H26
Global Const KEY_RIGHT = &H27
Global Const KEY_DOWN = &H28
Global Const KEY_SELECT = &H29
Global Const KEY_PRINT = &H2A
Global Const KEY_EXECUTE = &H2B
Global Const KEY_SNAPSHOT = &H2C
Global Const KEY_INSERT = &H2D
Global Const KEY_DELETE = &H2E
Global Const KEY_HELP = &H2F

' KEY_A thru KEY_Z are the same as their ASCII equivalents: 'A' thru 'Z'
' KEY_0 thru KEY_9 are the same as their ASCII equivalents: '0' thru '9'

Global Const KEY_NUMPAD0 = &H60
Global Const KEY_NUMPAD1 = &H61
Global Const KEY_NUMPAD2 = &H62
Global Const KEY_NUMPAD3 = &H63
Global Const KEY_NUMPAD4 = &H64
Global Const KEY_NUMPAD5 = &H65
Global Const KEY_NUMPAD6 = &H66
Global Const KEY_NUMPAD7 = &H67
Global Const KEY_NUMPAD8 = &H68
Global Const KEY_NUMPAD9 = &H69
Global Const KEY_MULTIPLY = &H6A
Global Const KEY_ADD = &H6B
Global Const KEY_SEPARATOR = &H6C
Global Const KEY_SUBTRACT = &H6D
Global Const KEY_DECIMAL = &H6E
Global Const KEY_DIVIDE = &H6F
Global Const KEY_F1 = &H70
Global Const KEY_F2 = &H71
Global Const KEY_F3 = &H72
Global Const KEY_F4 = &H73
Global Const KEY_F5 = &H74
Global Const KEY_F6 = &H75
```

```
Global Const KEY_F7 = &H76
Global Const KEY_F8 = &H77
Global Const KEY_F9 = &H78
Global Const KEY_F10 = &H79
Global Const KEY_F11 = &H7A
Global Const KEY_F12 = &H7B
Global Const KEY_F13 = &H7C
Global Const KEY_F14 = &H7D
Global Const KEY_F15 = &H7E
Global Const KEY_F16 = &H7F

Global Const KEY_NUMLOCK = &H90

' Variant VarType tags

Global Const V_EMPTY = 0
Global Const V_NULL = 1
Global Const V_INTEGER = 2
Global Const V_LONG = 3
Global Const V_SINGLE = 4
Global Const V_DOUBLE = 5
Global Const V_CURRENCY = 6
Global Const V_DATE = 7
Global Const V_STRING = 8


' Event Parameters

' ErrNum (LinkError)
Global Const WRONG_FORMAT = 1
Global Const DDE_SOURCE_CLOSED = 6
Global Const TOO_MANY_LINKS = 7
Global Const DATA_TRANSFER_FAILED = 8

' QueryUnload
Global Const FORM_CONTROLMENU = 0
Global Const FORM_CODE = 1
Global Const APP_WINDOWS = 2
Global Const APP_TASKMANAGER = 3
Global Const FORM_MDIFORM = 4

' Properties

' Colors
Global Const BLACK = &H0&
Global Const RED = &HFF&
Global Const GREEN = &HFF00&
Global Const YELLOW = &HFFFF&
Global Const BLUE = &HFF0000
Global Const MAGENTA = &HFF00FF
Global Const CYAN = &HFFFF00
Global Const WHITE = &HFFFFFF

' System Colors
Global Const SCROLL_BARS = &H80000000           ' Scroll-bars gray area.
Global Const DESKTOP = &H80000001               ' Desktop.
Global Const ACTIVE_TITLE_BAR = &H80000002      ' Active window caption.
Global Const INACTIVE_TITLE_BAR = &H80000003    ' Inactive window caption.
Global Const MENU_BAR = &H80000004              ' Menu background.
Global Const WINDOW_BACKGROUND = &H80000005     ' Window background.
Global Const WINDOW_FRAME = &H80000006          ' Window frame.
Global Const MENU_TEXT = &H80000007             ' Text in menus.
Global Const WINDOW_TEXT = &H80000008           ' Text in windows.
Global Const TITLE_BAR_TEXT = &H80000009        ' Text in caption, size box, scroll-bar arrow
box..
Global Const ACTIVE_BORDER = &H8000000A         ' Active window border.
Global Const INACTIVE_BORDER = &H8000000B       ' Inactive window border.
Global Const APPLICATION_WORKSPACE = &H8000000C ' Background color of multiple document interface
(MDI) applications.
Global Const HIGHLIGHT = &H8000000D             ' Items selected item in a control.
Global Const HIGHLIGHT_TEXT = &H8000000E        ' Text of item selected in a control.
Global Const BUTTON_FACE = &H8000000F           ' Face shading on command buttons.
Global Const BUTTON_SHADOW = &H80000010         ' Edge shading on command buttons.
Global Const GRAY_TEXT = &H80000011             ' Grayed (disabled) text.  This color is set to 0
```

```
if the current display driver does not support a solid gray color.
Global Const BUTTON_TEXT = &H80000012           ' Text on push buttons.

' Enumerated Types

' Align (picture box)
Global Const NONE = 0
Global Const ALIGN_TOP = 1
Global Const ALIGN_BOTTOM = 2

' Alignment
Global Const LEFT_JUSTIFY = 0  ' 0 - Left Justify
Global Const RIGHT_JUSTIFY = 1 ' 1 - Right Justify
Global Const CENTER = 2        ' 2 - Center

' BorderStyle (form)
'Global Const NONE = 0          ' 0 - None
Global Const FIXED_SINGLE = 1   ' 1 - Fixed Single
Global Const SIZABLE = 2        ' 2 - Sizable (Forms only)
Global Const FIXED_DOUBLE = 3   ' 3 - Fixed Double (Forms only)

' BorderStyle (Shape and Line)
'Global Const TRANSPARENT = 0    '0 - Transparent
'Global Const SOLID = 1          '1 - Solid
'Global Const DASH = 2          ' 2 - Dash
'Global Const DOT = 3           ' 3 - Dot
'Global Const DASH_DOT = 4      ' 4 - Dash-Dot
'Global Const DASH_DOT_DOT = 5 ' 5 - Dash-Dot-Dot
'Global Const INSIDE_SOLID = 6 ' 6 - Inside Solid

' MousePointer
Global Const DEFAULT = 0        ' 0 - Default
Global Const ARROW = 1          ' 1 - Arrow
Global Const CROSSHAIR = 2      ' 2 - Cross
Global Const IBEAM = 3          ' 3 - I-Beam
Global Const ICON_POINTER = 4   ' 4 - Icon
Global Const SIZE_POINTER = 5   ' 5 - Size
Global Const SIZE_NE_SW = 6     ' 6 - Size NE SW
Global Const SIZE_N_S = 7       ' 7 - Size N S
Global Const SIZE_NW_SE = 8     ' 8 - Size NW SE
Global Const SIZE_W_E = 9       ' 9 - Size W E
Global Const UP_ARROW = 10      ' 10 - Up Arrow
Global Const HOURGLASS = 11     ' 11 - Hourglass
Global Const NO_DROP = 12       ' 12 - No drop

' DragMode
Global Const MANUAL = 0    ' 0 - Manual
Global Const AUTOMATIC = 1 ' 1 - Automatic

' DrawMode
Global Const BLACKNESS = 1      ' 1 - Blackness
Global Const NOT_MERGE_PEN = 2  ' 2 - Not Merge Pen
Global Const MASK_NOT_PEN = 3   ' 3 - Mask Not Pen
Global Const NOT_COPY_PEN = 4   ' 4 - Not Copy Pen
Global Const MASK_PEN_NOT = 5   ' 5 - Mask Pen Not
Global Const INVERT = 6         ' 6 - Invert
Global Const XOR_PEN = 7        ' 7 - Xor Pen
Global Const NOT_MASK_PEN = 8   ' 8 - Not Mask Pen
Global Const MASK_PEN = 9       ' 9 - Mask Pen
Global Const NOT_XOR_PEN = 10   ' 10 - Not Xor Pen
Global Const NOP = 11           ' 11 - Nop
Global Const MERGE_NOT_PEN = 12 ' 12 - Merge Not Pen
Global Const COPY_PEN = 13      ' 13 - Copy Pen
Global Const MERGE_PEN_NOT = 14 ' 14 - Merge Pen Not
Global Const MERGE_PEN = 15     ' 15 - Merge Pen
Global Const WHITENESS = 16     ' 16 - Whiteness

' DrawStyle
Global Const SOLID = 0        ' 0 - Solid
Global Const DASH = 1         ' 1 - Dash
Global Const DOT = 2          ' 2 - Dot
Global Const DASH_DOT = 3     ' 3 - Dash-Dot
Global Const DASH_DOT_DOT = 4 ' 4 - Dash-Dot-Dot
Global Const INVISIBLE = 5    ' 5 - Invisible
```

```
Global Const INSIDE_SOLID = 6 ' 6 - Inside Solid

' FillStyle
' Global Const SOLID = 0           ' 0 - Solid
Global Const TRANSPARENT = 1       ' 1 - Transparent
Global Const HORIZONTAL_LINE = 2   ' 2 - Horizontal Line
Global Const VERTICAL_LINE = 3     ' 3 - Vertical Line
Global Const UPWARD_DIAGONAL = 4   ' 4 - Upward Diagonal
Global Const DOWNWARD_DIAGONAL = 5 ' 5 - Downward Diagonal
Global Const CROSS = 6             ' 6 - Cross
Global Const DIAGONAL_CROSS = 7    ' 7 - Diagonal Cross

' LinkMode (forms and controls)
' Global Const NONE = 0          ' 0 - None
Global Const LINK_SOURCE = 1     ' 1 - Source (forms only)
Global Const LINK_AUTOMATIC = 1  ' 1 - Automatic (controls only)
Global Const LINK_MANUAL = 2     ' 2 - Manual (controls only)
Global Const LINK_NOTIFY = 3     ' 3 - Notify (controls only)

' LinkMode (kept for VB1.0 compatibility, use new constants instead)
Global Const HOT = 1     ' 1 - Hot (controls only)
Global Const SERVER = 1  ' 1 - Server (forms only)
Global Const COLD = 2    ' 2 - Cold (controls only)


' ScaleMode
Global Const USER = 0         ' 0 - User
Global Const TWIPS = 1        ' 1 - Twip
Global Const POINTS = 2       ' 2 - Point
Global Const PIXELS = 3       ' 3 - Pixel
Global Const CHARACTERS = 4   ' 4 - Character
Global Const INCHES = 5       ' 5 - Inch
Global Const MILLIMETERS = 6  ' 6 - Millimeter
Global Const CENTIMETERS = 7  ' 7 - Centimeter

' ScrollBar
' Global Const NONE     = 0  ' 0 - None
Global Const HORIZONTAL = 1  ' 1 - Horizontal
Global Const VERTICAL = 2    ' 2 - Vertical
Global Const BOTH = 3        ' 3 - Both

' Shape
Global Const SHAPE_RECTANGLE = 0
Global Const SHAPE_SQUARE = 1
Global Const SHAPE_OVAL = 2
Global Const SHAPE_CIRCLE = 3
Global Const SHAPE_ROUNDED_RECTANGLE = 4
Global Const SHAPE_ROUNDED_SQUARE = 5

' WindowState
Global Const NORMAL = 0     ' 0 - Normal
Global Const MINIMIZED = 1  ' 1 - Minimized
Global Const MAXIMIZED = 2  ' 2 - Maximized

' Check Value
Global Const UNCHECKED = 0  ' 0 - Unchecked
Global Const CHECKED = 1    ' 1 - Checked
Global Const GRAYED = 2     ' 2 - Grayed

' Shift parameter masks
Global Const SHIFT_MASK = 1
Global Const CTRL_MASK = 2
Global Const ALT_MASK = 4

' Button parameter masks
Global Const LEFT_BUTTON = 1
Global Const RIGHT_BUTTON = 2
Global Const MIDDLE_BUTTON = 4

' Function Parameters
' MsgBox parameters
Global Const MB_OK = 0                  ' OK button only
Global Const MB_OKCANCEL = 1            ' OK and Cancel buttons
Global Const MB_ABORTRETRYIGNORE = 2    ' Abort, Retry, and Ignore buttons
```

```
Global Const MB_YESNOCANCEL = 3          ' Yes, No, and Cancel buttons
Global Const MB_YESNO = 4                ' Yes and No buttons
Global Const MB_RETRYCANCEL = 5          ' Retry and Cancel buttons

Global Const MB_ICONSTOP = 16            ' Critical message
Global Const MB_ICONQUESTION = 32        ' Warning query
Global Const MB_ICONEXCLAMATION = 48     ' Warning message
Global Const MB_ICONINFORMATION = 64     ' Information message

Global Const MB_APPLMODAL = 0            ' Application Modal Message Box
Global Const MB_DEFBUTTON1 = 0           ' First button is default
Global Const MB_DEFBUTTON2 = 256         ' Second button is default
Global Const MB_DEFBUTTON3 = 512         ' Third button is default
Global Const MB_SYSTEMMODAL = 4096       'System Modal

' MsgBox return values
Global Const IDOK = 1                    ' OK button pressed
Global Const IDCANCEL = 2                ' Cancel button pressed
Global Const IDABORT = 3                 ' Abort button pressed
Global Const IDRETRY = 4                 ' Retry button pressed
Global Const IDIGNORE = 5                ' Ignore button pressed
Global Const IDYES = 6                   ' Yes button pressed
Global Const IDNO = 7                    ' No button pressed

' SetAttr, Dir, GetAttr functions
Global Const ATTR_NORMAL = 0
Global Const ATTR_READONLY = 1
Global Const ATTR_HIDDEN = 2
Global Const ATTR_SYSTEM = 4
Global Const ATTR_VOLUME = 8
Global Const ATTR_DIRECTORY = 16
Global Const ATTR_ARCHIVE = 32

'Grid
'ColAlignment,FixedAlignment Properties
Global Const GRID_ALIGNLEFT = 0
Global Const GRID_ALIGNRIGHT = 1
Global Const GRID_ALIGNCENTER = 2

'Fillstyle Property
Global Const GRID_SINGLE = 0
Global Const GRID_REPEAT = 1

'OLE Client Control
'Action
Global Const OLE_CREATE_NEW = 0
Global Const OLE_CREATE_FROM_FILE = 1
Global Const OLE_COPY = 4
Global Const OLE_PASTE = 5
Global Const OLE_UPDATE = 6
Global Const OLE_ACTIVATE = 7
Global Const OLE_EXECUTE = 8
Global Const OLE_CLOSE = 9
Global Const OLE_DELETE = 10
Global Const OLE_SAVE_TO_FILE = 11
Global Const OLE_READ_FROM_FILE = 12
Global Const OLE_CONVERT_TO_TYPE = 13

'ServerType
Global Const OLE_LINKED = 0
Global Const OLE_EMBEDDED = 1
Global Const OLE_STATIC = 2

'UpdateOptions
Global Const OLE_AUTOMATIC = 0
Global Const OLE_FROZEN = 1
Global Const OLE_MANUAL = 2

'Update Event Constants
Global Const OLE_CHANGED = 0
Global Const OLE_SAVED = 1
Global Const OLE_CLOSED = 2
Global Const OLE_RELEASE = 3
```

```
'-------------------------------------------------------
'      Table of Contents for Visual Basic Professional
'
'       1.  3-D Controls
'           (Frame/Panel/Option/Check/Command/Group Push)
'       2.  Animated Button
'       3.  Common Dialog Section
'       4.  Gauge Control
'       5.  Graph Control Section
'       6.  Key Status Control
'       7.  Spin Button
'       8.  MCI Control (Multimedia)
'       9.  Masked Edit Control
'       10. Comm Control
'       11. ODBC Constants
'-------------------------------------------------------

'---------------------------------
'Common Dialog Control
'---------------------------------
'Action Property
Global Const DLG_FILE_OPEN = 1
Global Const DLG_FILE_SAVE = 2
Global Const DLG_COLOR = 3
Global Const DLG_FONT = 4
Global Const DLG_PRINT = 5
Global Const DLG_HELP = 6

'File Open/Save Dialog Flags
Global Const OFN_READONLY = &H1&
Global Const OFN_OVERWRITEPROMPT = &H2&
Global Const OFN_HIDEREADONLY = &H4&
Global Const OFN_NOCHANGEDIR = &H8&
Global Const OFN_SHOWHELP = &H10&
Global Const OFN_NOVALIDATE = &H100&
Global Const OFN_ALLOWMULTISELECT = &H200&
Global Const OFN_EXTENSIONDIFFERENT = &H400&
Global Const OFN_PATHMUSTEXIST = &H800&
Global Const OFN_FILEMUSTEXIST = &H1000&
Global Const OFN_CREATEPROMPT = &H2000&
Global Const OFN_SHAREAWARE = &H4000&
Global Const OFN_NOREADONLYRETURN = &H8000&

'Color Dialog Flags
Global Const CC_RGBINIT = &H1&
Global Const CC_FULLOPEN = &H2&
Global Const CC_PREVENTFULLOPEN = &H4&
Global Const CC_SHOWHELP = &H8&

'Fonts Dialog Flags
Global Const CF_SCREENFONTS = &H1&
Global Const CF_PRINTERFONTS = &H2&
Global Const CF_BOTH = &H3&
Global Const CF_SHOWHELP = &H4&
Global Const CF_INITTOLOGFONTSTRUCT = &H40&
Global Const CF_USESTYLE = &H80&
Global Const CF_EFFECTS = &H100&
Global Const CF_APPLY = &H200&
Global Const CF_ANSIONLY = &H400&
Global Const CF_NOVECTORFONTS = &H800&
Global Const CF_NOSIMULATIONS = &H1000&
Global Const CF_LIMITSIZE = &H2000&
Global Const CF_FIXEDPITCHONLY = &H4000&
Global Const CF_WYSIWYG = &H8000&                    'must also have CF_SCREENFONTS &
CF_PRINTERFONTS
Global Const CF_FORCEFONTEXIST = &H10000
Global Const CF_SCALABLEONLY = &H20000
Global Const CF_TTONLY = &H40000
Global Const CF_NOFACESEL = &H80000
Global Const CF_NOSTYLESEL = &H100000
Global Const CF_NOSIZESEL = &H200000

'Printer Dialog Flags
Global Const PD_ALLPAGES = &H0&
```

```
Global Const PD_SELECTION = &H1&
Global Const PD_PAGENUMS = &H2&
Global Const PD_NOSELECTION = &H4&
Global Const PD_NOPAGENUMS = &H8&
Global Const PD_COLLATE = &H10&
Global Const PD_PRINTTOFILE = &H20&
Global Const PD_PRINTSETUP = &H40&
Global Const PD_NOWARNING = &H80&
Global Const PD_RETURNDC = &H100&
Global Const PD_RETURNIC = &H200&
Global Const PD_RETURNDEFAULT = &H400&
Global Const PD_SHOWHELP = &H800&
Global Const PD_USEDEVMODECOPIES = &H40000
Global Const PD_DISABLEPRINTTOFILE = &H80000
Global Const PD_HIDEPRINTTOFILE = &H100000

'Help Constants
Global Const HELP_CONTEXT = &H1          'Display topic in ulTopic
Global Const HELP_QUIT = &H2             'Terminate help
Global Const HELP_INDEX = &H3            'Display index
Global Const HELP_CONTENTS = &H3
Global Const HELP_HELPONHELP = &H4       'Display help on using help
Global Const HELP_SETINDEX = &H5         'Set the current Index for multi index help
Global Const HELP_SETCONTENTS = &H5
Global Const HELP_CONTEXTPOPUP = &H8
Global Const HELP_FORCEFILE = &H9
Global Const HELP_KEY = &H101            'Display topic for keyword in offabData
Global Const HELP_COMMAND = &H102
Global Const HELP_PARTIALKEY = &H105     'call the search engine in winhelp

'Error Constants
Global Const CDERR_DIALOGFAILURE = &HFFFF

Global Const CDERR_GENERALCODES = &H0
Global Const CDERR_STRUCTSIZE = &H1
Global Const CDERR_INITIALIZATION = &H2
Global Const CDERR_NOTEMPLATE = &H3
Global Const CDERR_NOHINSTANCE = &H4
Global Const CDERR_LOADSTRFAILURE = &H5
Global Const CDERR_FINDRESFAILURE = &H6
Global Const CDERR_LOADRESFAILURE = &H7
Global Const CDERR_LOCKRESFAILURE = &H8
Global Const CDERR_MEMALLOCFAILURE = &H9
Global Const CDERR_MEMLOCKFAILURE = &HA
Global Const CDERR_NOHOOK = &HB

'Added for CMDLG.VBX
Global Const CDERR_CANCEL = &HC
Global Const CDERR_NODLL = &HD
Global Const CDERR_ERRPROC = &HE
Global Const CDERR_ALLOC = &HF
Global Const CDERR_HELP = &H10

Global Const PDERR_PRINTERCODES = &H1000
Global Const PDERR_SETUPFAILURE = &H1001
Global Const PDERR_PARSEFAILURE = &H1002
Global Const PDERR_RETDEFFAILURE = &H1003
Global Const PDERR_LOADDRVFAILURE = &H1004
Global Const PDERR_GETDEVMODEFAIL = &H1005
Global Const PDERR_INITFAILURE = &H1006
Global Const PDERR_NODEVICES = &H1007
Global Const PDERR_NODEFAULTPRN = &H1008
Global Const PDERR_DNDMMISMATCH = &H1009
Global Const PDERR_CREATEICFAILURE = &H100A
Global Const PDERR_PRINTERNOTFOUND = &H100B

Global Const CFERR_CHOOSEFONTCODES = &H2000
Global Const CFERR_NOFONTS = &H2001

Global Const FNERR_FILENAMECODES = &H3000
Global Const FNERR_SUBCLASSFAILURE = &H3001
Global Const FNERR_INVALIDFILENAME = &H3002
Global Const FNERR_BUFFERTOOSMALL = &H3003
```

```
Global Const FRERR_FINDREPLACECODES = &H4000
Global Const CCERR_CHOOSECOLORCODES = &H5000

'--------------------------------------
'Graph Control
'--------------------------------------
'General
Global Const G_NONE = 0
Global Const G_DEFAULT = 0

Global Const G_OFF = 0
Global Const G_ON = 1

Global Const G_MONO = 0
Global Const G_COLOR = 1

'Graph Types
Global Const G_PIE2D = 1
Global Const G_PIE3D = 2
Global Const G_BAR2D = 3
Global Const G_BAR3D = 4
Global Const G_GANTT = 5
Global Const G_LINE = 6
Global Const G_LOGLIN = 7
Global Const G_AREA = 8
Global Const G_SCATTER = 9
Global Const G_POLAR = 10
Global Const G_HLC = 11

'Colors
Global Const G_BLACK = 0
Global Const G_BLUE = 1
Global Const G_GREEN = 2
Global Const G_CYAN = 3
Global Const G_RED = 4
Global Const G_MAGENTA = 5
Global Const G_BROWN = 6
Global Const G_LIGHT_GRAY = 7
Global Const G_DARK_GRAY = 8
Global Const G_LIGHT_BLUE = 9
Global Const G_LIGHT_GREEN = 10
Global Const G_LIGHT_CYAN = 11
Global Const G_LIGHT_RED = 12
Global Const G_LIGHT_MAGENTA = 13
Global Const G_YELLOW = 14
Global Const G_WHITE = 15
Global Const G_AUTOBW = 16

'Patterns
Global Const G_SOLID = 0
Global Const G_HOLLOW = 1
Global Const G_HATCH1 = 2
Global Const G_HATCH2 = 3
Global Const G_HATCH3 = 4
Global Const G_HATCH4 = 5
Global Const G_HATCH5 = 6
Global Const G_HATCH6 = 7
Global Const G_BITMAP1 = 16
Global Const G_BITMAP2 = 17
Global Const G_BITMAP3 = 18
Global Const G_BITMAP4 = 19
Global Const G_BITMAP5 = 20
Global Const G_BITMAP6 = 21
Global Const G_BITMAP7 = 22
Global Const G_BITMAP8 = 23
Global Const G_BITMAP9 = 24
Global Const G_BITMAP10 = 25
Global Const G_BITMAP11 = 26
Global Const G_BITMAP12 = 27
Global Const G_BITMAP13 = 28
Global Const G_BITMAP14 = 29
Global Const G_BITMAP15 = 30
Global Const G_BITMAP16 = 31
```

```
'Symbols
Global Const G_CROSS_PLUS = 0
Global Const G_CROSS_TIMES = 1
Global Const G_TRIANGLE_UP = 2
Global Const G_SOLID_TRIANGLE_UP = 3
Global Const G_TRIANGLE_DOWN = 4
Global Const G_SOLID_TRIANGLE_DOWN = 5
Global Const G_SQUARE = 6
Global Const G_SOLID_SQUARE = 7
Global Const G_DIAMOND = 8
Global Const G_SOLID_DIAMOND = 9

'Line Styles
'Global Const G_SOLID = 0
Global Const G_DASH = 1
Global Const G_DOT = 2
Global Const G_DASHDOT = 3
Global Const G_DASHDOTDOT = 4

'Grids
Global Const G_HORIZONTAL = 1
Global Const G_VERTICAL = 2

'Statistics
Global Const G_MEAN = 1
Global Const G_MIN_MAX = 2
Global Const G_STD_DEV = 4
Global Const G_BEST_FIT = 8

'Data Arrays
Global Const G_GRAPH_DATA = 1
Global Const G_COLOR_DATA = 2
Global Const G_EXTRA_DATA = 3
Global Const G_LABEL_TEXT = 4
Global Const G_LEGEND_TEXT = 5
Global Const G_PATTERN_DATA = 6
Global Const G_SYMBOL_DATA = 7
Global Const G_XPOS_DATA = 8
Global Const G_ALL_DATA = 9

'Draw Mode
Global Const G_NO_ACTION = 0
Global Const G_CLEAR = 1
Global Const G_DRAW = 2
Global Const G_BLIT = 3
Global Const G_COPY = 4
Global Const G_PRINT = 5
Global Const G_WRITE = 6

'Print Options
Global Const G_BORDER = 2

'Pie Chart Options                '
Global Const G_NO_LINES = 1
Global Const G_COLORED = 2
Global Const G_PERCENTS = 4

'Bar Chart Options                '
'Global Const G_HORIZONTAL = 1
Global Const G_STACKED = 2
Global Const G_PERCENTAGE = 4
Global Const G_Z_CLUSTERED = 6

'Gantt Chart Options              '
Global Const G_SPACED_BARS = 1

'Line/Polar Chart Options         '
Global Const G_SYMBOLS = 1
Global Const G_STICKS = 2
Global Const G_LINES = 4

'Area Chart Options               '
Global Const G_ABSOLUTE = 1
Global Const G_PERCENT = 2
```

```
'HLC Chart Options                '
Global Const G_NO_CLOSE = 1
Global Const G_NO_HIGH_LOW = 2


' Agility/VB trappable error codes
' Updated: 07-Sep-92

Global Const AGIE_DBOPEN = 4100
Global Const AGIE_DBNOTOPEN = 4101
Global Const AGIE_NOTEXIST = 4102
Global Const AGIE_NOUPDATE = 4103
Global Const AGIE_NODELETE = 4104
Global Const AGIE_NOADD = 4105
Global Const AGIE_BADFILE = 4106
Global Const AGIE_NOCURPOS = 4107
Global Const AGIE_BOUNDS = 4108
Global Const AGIE_ROWNUM = 4109
Global Const AGIE_BADSORT = 4110
Global Const AGIE_BADQUERY = 4111
Global Const AGIE_BADPARAM = 4112
Global Const AGIE_NOPERMIT = 4113
Global Const AGIE_MARKERR = 4114
Global Const AGIE_SETERR = 4115
Global Const AGIE_RECACCESS = 4116
Global Const AGIE_FLDACCESS = 4117
Global Const AGIE_INDEXERR = 4118

Function NDEV (Mean, SD) As Single
    Const Pi = 3.1415926536
    NDEV = Sqr(2 * Log(1 / Rnd)) * Cos(2 * Pi * Rnd) * SD + Mean
End Function

Function NL () As String
  NL = Chr$(10) + Chr$(13)
End Function
```

Code For Agi.bas - database

```
' Agility/VB Release 001 (0.243) Definitions
' Copyright (C) Apex Software Corporation, 1992.  All rights reserved.
'
' ** REQUIRES VISUAL BASIC 2.0 ***
'

' File Maintenance

Declare Sub AgiViewOpen Lib "agivb001.dll" (ByVal hand%, ByVal viewname$, ByVal mode$)
Declare Sub AgiSchemaOpen Lib "agivb001.dll" (ByVal hand%, ByVal dbhand%)
Declare Function AgiFreeFile Lib "agivb001.dll" () As Integer
Declare Sub AgiViewClose Lib "agivb001.dll" (ByVal hand%)
Declare Sub AgiViewCloseAll Lib "agivb001.dll" ()
Declare Function AgiError Lib "agivb001.dll" () As Integer
Declare Function AgiErrorText Lib "agivb001.dll" () As String
Declare Function AgiInternalError Lib "agivb001.dll" () As Integer
Declare Function AgiVersion Lib "agivb001.dll" () As String

' Record maintenance

Declare Sub AgiViewDelete Lib "agivb001.dll" (ByVal hand%)

' Record data access and manipulation

Declare Sub AgiViewAdd Lib "agivb001.dll" (ByVal hand%, ByVal defs$, lpRec As Any)
Declare Sub AgiViewGet Lib "agivb001.dll" (ByVal hand%, ByVal defs$, lpRec As Any)
Declare Sub AgiViewUpdate Lib "agivb001.dll" (ByVal hand%, ByVal defs$, lpRec As Any)
Declare Function AgiViewCount Lib "agivb001.dll" (ByVal hand%) As Long

' Ordering

Declare Sub AgiViewSort Lib "agivb001.dll" (ByVal hand%, ByVal sorts$)
Declare Sub AgiViewUnsort Lib "agivb001.dll" (ByVal hand%)
Declare Function AgiViewSortedOn Lib "agivb001.dll" (ByVal hand%) As String

' Positioning

Declare Function AgiViewFirst Lib "agivb001.dll" (ByVal hand%) As Integer
Declare Function AgiViewNext Lib "agivb001.dll" (ByVal hand%) As Integer
Declare Function AgiViewLast Lib "agivb001.dll" (ByVal hand%) As Integer
Declare Function AgiViewPrevious Lib "agivb001.dll" (ByVal hand%) As Integer
Declare Function AgiViewGetRow Lib "agivb001.dll" (ByVal hand%) As Long
Declare Sub AgiViewSetRow Lib "agivb001.dll" (ByVal hand%, ByVal rownum&)

' Set Selection

Declare Sub AgiViewFind Lib "agivb001.dll" (ByVal hand%, ByVal defs$, lpRec As Any)
Declare Sub AgiViewFindAlso Lib "agivb001.dll" (ByVal hand%, ByVal defs$, lpRec As Any)
Declare Sub AgiViewFindAll Lib "agivb001.dll" (ByVal hand%)
Declare Sub AgiViewRefind Lib "agivb001.dll" (ByVal hand%)
Declare Sub AgiViewEmptySet Lib "agivb001.dll" (ByVal hand%)

' Set save, restore, copy and deletion

Declare Sub AgiViewMemorizeSet Lib "agivb001.dll" (ByVal hand%, ByVal setname$)
Declare Sub AgiViewRecallSet Lib "agivb001.dll" (ByVal hand%, ByVal setname$)
Declare Sub AgiViewForgetSet Lib "agivb001.dll" (ByVal hand%, ByVal setname$)
Declare Sub AgiViewCopySet Lib "agivb001.dll" (ByVal srch%, ByVal desth%)
Declare Sub AgiViewDeleteSet Lib "agivb001.dll" (ByVal hand%)

Declare Sub AgiViewCopyRecord Lib "agivb001.dll" (ByVal srch%, ByVal desth%)

' Mark manipulation

Declare Sub AgiViewMark Lib "agivb001.dll" (ByVal hand%, ByVal markname$)
Declare Sub AgiViewToMark Lib "agivb001.dll" (ByVal hand%, ByVal markname$)
Declare Sub AgiViewForgetMark Lib "agivb001.dll" (ByVal hand%, ByVal markname$)

' Record accumulation

Declare Sub AgiViewBuildRecord Lib "agivb001.dll" (ByVal hand%, ByVal defs$, lpRec As Any)
Declare Function AgiGatherData Lib "agivb001.dll" (ByVal hand%, frm As Form) As Integer
```

```
Declare Function AgiGatherCtlData Lib "agivb001.dll" (ByVal hand%, ctl As Control) As Integer

' Special definitions to support array storage

Declare Sub AgiViewGetArray Lib "agivb001.dll" Alias "AgiViewGet" (ByVal hand%, ByVal defs$,
lpRec() As Any)
Declare Sub AgiViewUpdateArray Lib "agivb001.dll" Alias "AgiViewUpdate" (ByVal hand%, ByVal defs$,
lpRec() As Any)
Declare Sub AgiViewAddArray Lib "agivb001.dll" Alias "AgiViewAdd" (ByVal hand%, ByVal defs$,
lpRec() As Any)
Declare Sub AgiViewBuildRecordArray Lib "agivb001.dll" Alias "AgiViewBuildRecord" (ByVal hand%,
ByVal defs$, lpRec() As Any)
```

Code for BehmStat Form

```
Option Explicit

Sub cmdCalc_Click ()
    Dim Mode        As String
    Dim RecMode     As String
    Dim ModeCount   As Integer
    Dim RecCount    As Integer

    Dim PI              As Double
    Dim BaseCoefficient As Single
    Dim BaseExponent    As Single
    Dim PredCoefficient As Single
    Dim PredExponent    As Single
    Dim EstEffort       As Double
    Dim LogEstEffort    As Double
    Dim ACTEFFORT       As Single
    Dim LogActEffort    As Double
    Dim ActEffMean      As Double
    Dim LogActEffMean   As Double

    Dim BoehmHandle As Single

    Dim SumActEff   As Double
    Dim SumLogActEff As Double
    Dim SSE         As Double
    Dim LogSSE      As Double
    Dim LogSSTO     As Double
    Dim SumMRE      As Double
    Dim SumLogMRE   As Double
    Dim R2          As Single
    Dim RRMS        As Single
    Dim MRE         As Single
    Dim LogMRE      As Single
    Dim MREMean     As Single
    Dim PredLevel   As Single
    Const MRELimit = .25
    Dim ImproveLevel As Single

    If optOrganic.Value = True Then
        Mode = "ORG"
    ElseIf optSemiD.Value = True Then
        Mode = "SD"
    ElseIf optEmbedded.Value = True Then
        Mode = "E"
    ElseIf optAllData.Value = True Then
        Mode = "ALL"
    Else
        Mode = "UNK"
    End If
    Mode = Trim(Mode)

    BaseCoefficient = Val(txtBaseCoeff.Text)
    BaseExponent = Val(txtBaseExp.Text)
    PredCoefficient = Val(txtPredCoeff.Text)
    PredExponent = Val(txtPredExp.Text)

    BoehmHandle = DBHandle
    'AgiViewOpen BoehmHandle, "boehm's.dbf", "R"
    'Debug.Print AgiErrorText()
    more = AgiViewFirst(BoehmHandle)

    'Find Average of Actual Efforts in Real and Log domains
    SumActEff = 0#
    SumLogActEff = 0#
    ModeCount = 0

    Do While more
        AgiViewGet BoehmHandle, "MODE(S)", RecMode
        If Mode = Trim(RecMode) Or Mode = "ALL" Then
            ModeCount = ModeCount + 1
            'txtNumMode.Text = Format$(ModeCount, "###0")
            AgiViewGet BoehmHandle, "ACTEFFORT(F)", ACTEFFORT
```

```
            SumActEff = SumActEff + ACTEFFORT
            SumLogActEff = SumLogActEff + Log(ACTEFFORT)
        End If
        more = AgiViewNext(BoehmHandle)
    Loop
    If ModeCount > 0 Then
        ActEffMean = SumActEff / ModeCount
        LogActEffMean = SumLogActEff / ModeCount
    Else
        MsgBox "No Records found with matching Mode."
        Exit Sub
    End If

    'Caluclate Statistics
    SSE = 0#
    LogSSE = 0#
    LogSSTO = 0#
    SumMRE = 0#
    SumLogMRE = 0#
    PredLevel = 0!
    ImproveLevel = 0!
    RecCount = 0
    ModeCount = 0

    more = AgiViewFirst(BoehmHandle)
    txtNumMode.Text = Format$(ModeCount, "###0")
    Do While more
        RecCount = RecCount + 1
        txtNumRec.Text = Format$(RecCount, "###0")
        AgiViewGet BoehmHandle, BoehmRDS, BoehmRec
        If Mode = Trim(BoehmRec.Mode) Or Mode = "ALL" Then
            ModeCount = ModeCount + 1
            txtNumMode.Text = Format$(ModeCount, "###0")
            PI = BoehmRec.RELY * BoehmRec.DBSIZE * BoehmRec.CPLX * BoehmRec.EXTIME
            PI = PI * BoehmRec.STOR * BoehmRec.VIRT * BoehmRec.TURN * BoehmRec.ACAP
            PI = PI * BoehmRec.AEXP * BoehmRec.PCAP * BoehmRec.VEXP * BoehmRec.LEXP
            PI = PI * BoehmRec.MODP * BoehmRec.TOOL * BoehmRec.SCED * BoehmRec.RVOL
            EstEffort = PI * PredCoefficient * (BoehmRec.ADJKDSI ^ PredExponent)
            LogEstEffort = Log(EstEffort)
            MRE = Abs(BoehmRec.ACTEFFORT - EstEffort)
            LogMRE = Abs(Log(BoehmRec.ACTEFFORT) - LogEstEffort)

            'Calculate sums BoehmRec.STOR
            SSE = SSE + MRE ^ 2
            LogSSE = LogSSE + LogMRE ^ 2
            LogSSTO = LogSSTO + (Log(BoehmRec.ACTEFFORT) - LogActEffMean) ^ 2
            SumMRE = SumMRE + MRE
            If MRE / BoehmRec.ACTEFFORT <= MRELimit Then
                PredLevel = PredLevel + 1#
            End If
            If MRE < Abs(BoehmRec.ACTEFFORT - (PI * BaseCoefficient * (BoehmRec.ADJKDSI ^
BaseExponent))) Then
                ImproveLevel = ImproveLevel + 1#
            End If
        End If
        more = AgiViewNext(BoehmHandle)
    Loop
    'AgiViewClose BoehmHandle

    If ModeCount > 0 Then
        R2 = 1 - (LogSSE * (ModeCount - 1)) / (LogSSTO * (ModeCount - 2))
        RRMS = (Sqr(SSE / ModeCount)) / ActEffMean
        MREMean = SumMRE / ModeCount
        PredLevel = (PredLevel / ModeCount) * 100
        ImproveLevel = (ImproveLevel / ModeCount) * 100
    End If

    Text3.Text = Format$(R2, "0.0000")
    Text4.Text = Format$(RRMS, "0.000")
    Text5.Text = Format$(MREMean, "0.00")
    Text6.Text = Format$(PredLevel, "#0.0") + "%"
    Text9.Text = Format$(ImproveLevel, "#0.0") + "%"

End Sub
```

```
Sub cmdCancel_Click ()
    Unload frmStatBoehm
End Sub

Sub optEmbedded_Click ()
    txtBaseCoeff.Text = "2.8"
    txtBaseExp.Text = "1.20"
End Sub

Sub optOrganic_Click ()
    txtBaseCoeff.Text = "3.2"
    txtBaseExp = "1.05"
End Sub

Sub optSemiD_Click ()
    txtBaseCoeff.Text = "3.0"
    txtBaseExp.Text = "1.12"
End Sub
```

Code for BhmGraph Form

```
Option Explicit
Option Base 1

Sub cmdCancel_Click ()
    Graph1.DataReset = G_All_Data
    Unload frmBoehmGraph
End Sub

Sub cmdNorm_Click ()
    If NormState = Norm_Off Then
        cmdNorm.Caption = "Normalize Off"
        NormState = Norm_On
    Else
        cmdNorm.Caption = "Normalize On"
        NormState = Norm_Off
    End If

    Select Case DBType
        Case BoehmDB
            LoadBoehmData XMin, XMax, NormState
        Case SampleDB
            LoadSampleData XMin, XMax, NormState
        Case ThesisDB, CompThesisDB
        Case UnknownDB
    End Select

    Graph1.DrawMode = G_Draw
    If FileGraph = 1 Then
        Graph1.DrawMode = G_Write
    End If
End Sub

Sub cmdZoom_Click ()
    Dim TempYMax As Single
    Dim TempYMin As Single
    Dim TempXMax As Single
    Dim TempXMin As Single
    Dim Redraw As Integer

    Redraw = False

    TempYMax = YMax
    TempYMin = YMin
    TempXMax = XMax
    TempXMin = XMin

    frmZoomData.Show Modal

    If XMax <> TempXMax Or XMin <> TempXMin Then
        Redraw = True
        Graph1.DataReset = G_All_Data
        Select Case DBType
            Case BoehmDB
                LoadBoehmData XMin, XMax, NormState
            Case SampleDB
                LoadSampleData XMin, XMax, NormState
            Case ThesisDB, CompThesisDB
            Case UnknownDB
        End Select
    End If

    If YMax <> TempYMax Then
        Redraw = True
        Graph1.YAxisMax = YMax
    End If

    If YMin <> TempYMin Then
        Redraw = True
        Graph1.YAxisMin = YMin
    End If

    If Redraw Then
```

```
        Graph1.DrawMode = G_Draw
    End If
    If FileGraph = 1 Then
        Graph1.DrawMode = G_Write
    End If
End Sub

Sub Form_Load ()
    txtDBName.Text = DBFileSpec
    Screen.MousePointer = HourGlass

    'set filename for saving graph
    Graph1.ImageFile = "behmgrph"
    Graph1.NumPoints = 100
    Graph1.IndexStyle = 1   'Enhanced index style permits access to
                            'graph 2-D arrays for scatter data.

    Graph1.YAxisStyle = 2   'Permits user defined Y-origin through YAxisMin,
    Graph1.YAxisTicks = 8

    'Set default Axis limits for graph
    YMax = 1#       'Will increase upon reading database
    YMin = 0#
    XMax = 3000#    'May decrease to fit x range of data
    XMin = 0#

    WindowState = Maximized
    NormState = Norm_Off
    Graph1.LeftTitle = "MM"
    Graph1.BottomTitle = "Size in KDSI"
    Graph1.DrawMode = G_Draw

    Select Case DBType
        Case BoehmDB
            LoadBoehmData XMin, XMax, NormState
        Case SampleDB
            LoadSampleData XMin, XMax, NormState
        Case ThesisDB, CompThesisDB
        Case UnknownDB
    End Select
    Screen.MousePointer = HourGlass

    'Set Y-axis scale
    Graph1.YAxisMax = YMax
    Graph1.YAxisMin = YMin
    Graph1.DrawMode = G_Draw
    If FileGraph = 1 Then
        Graph1.DrawMode = G_Write
    End If
    If FileGraph = 1 Then
        Graph1.DrawMode = G_Write
    End If
    Screen.MousePointer = Default
End Sub

Sub LoadBoehmData (XLower As Single, XUpper As Single, Normalize As Integer)
    Dim Mode        As Integer
    Const NumModes = 3
    Graph1.NumSets = NumModes
    Dim I As Integer

    'Array subscripting
    '       1 = Organic Mode
    '       2 = Semidetached Mode
    '       3 = Embedded Mode

    ReDim ModeCount(NumModes)   As Integer

    Dim PI                      As Double
    ReDim Coefficient(NumModes) As Single
    ReDim Exponent(NumModes)    As Single
    ReDim CoeffOnly(NumModes)   As Single
    ReDim ExpOnly(NumModes)     As Single
    ExpOnly(1) = 1.05
```

```
ExpOnly(2) = 1.12
ExpOnly(3) = 1.2

Dim BoehmHandle             As Integer
Dim RecCount                As Integer

ReDim TempQ(NumModes)       As Double
ReDim SumMMQ(NumModes)      As Double
ReDim SumQ2(NumModes)       As Double
ReDim TempLog(NumModes)     As Double
ReDim TempDiv(NumModes)     As Double
ReDim a0(NumModes)          As Integer
ReDim a1(NumModes)          As Double
ReDim a2(NumModes)          As Double
ReDim d0(NumModes)          As Double
ReDim d1(NumModes)          As Double

'Intialize all mode arrays to zero
For I = 1 To NumModes
    SumMMQ(I) = 0#
    SumQ2(I) = 0#
    ModeCount(I) = 0
    a1(I) = 0#
    a2(I) = 0#
    d0(I) = 0#
    d1(I) = 0#
Next I
RecCount = 0

BoehmHandle = DBHandle
more = AgiViewFirst(BoehmHandle)
Screen.MousePointer = HourGlass
Do While more
    AgiViewGet BoehmHandle, BoehmRDS, BoehmRec

    RecCount = RecCount + 1

    'Set appropriate Mode and update correct ModeCount
    Select Case Trim(BoehmRec.Mode)
        Case "ORG"
            Mode = 1
        Case "SD"
            Mode = 2
        Case "E"
            Mode = 3
        Case Else
            MsgBox "Unrecognized Mode at record " + Str(RecCount) + "."
            Mode = 1
    End Select
    ModeCount(Mode) = ModeCount(Mode) + 1

    'Calculate the Muliplier, PI
    PI = BoehmRec.RELY * BoehmRec.DBSIZE * BoehmRec.CPLX * BoehmRec.EXTIME
    PI = PI * BoehmRec.STOR * BoehmRec.VIRT * BoehmRec.TURN * BoehmRec.ACAP
    PI = PI * BoehmRec.AEXP * BoehmRec.PCAP * BoehmRec.VEXP * BoehmRec.LEXP
    PI = PI * BoehmRec.MODP * BoehmRec.TOOL * BoehmRec.SCED * BoehmRec.RVOL

    'Calculate sums for Coefficient only
    TempQ(Mode) = PI * (BoehmRec.ADJKDSI ^ ExpOnly(Mode))
    SumMMQ(Mode) = SumMMQ(Mode) + BoehmRec.ACTEFFORT * TempQ(Mode)
    SumQ2(Mode) = SumQ2(Mode) + TempQ(Mode) * TempQ(Mode)

    'Calculate sums for Coefficient and Exponent
    TempLog(Mode) = Log(BoehmRec.ADJKDSI)
    TempDiv(Mode) = Log(BoehmRec.ACTEFFORT / PI)
    a1(Mode) = a1(Mode) + TempLog(Mode)
    a2(Mode) = a2(Mode) + TempLog(Mode) * TempLog(Mode)
    d0(Mode) = d0(Mode) + TempDiv(Mode)
    d1(Mode) = d1(Mode) + TempDiv(Mode) * TempLog(Mode)

    If BoehmRec.ADJKDSI >= XLower And BoehmRec.ADJKDSI <= XUpper Then
        'Load Data to the Graph
        Graph1.ThisSet = Mode
        Graph1.ThisPoint = ModeCount(Mode)
```

```
            If Normalize Then
                Graph1.GraphData = BoehmRec.ACTEFFORT / PI
            Else
                Graph1.GraphData = BoehmRec.ACTEFFORT
            End If
            Graph1.XPosData = BoehmRec.ADJKDSI

            'Check for maximum Effort and update if needed
            If YMax < BoehmRec.ACTEFFORT Then
                YMax = BoehmRec.ACTEFFORT
            End If
        End If

        more = AgiViewNext(BoehmHandle)
    Loop

    For I = 1 To NumModes
        'Calculate equation parameters
        CoeffOnly(I) = SumMMQ(I) / SumQ2(I)
        a0(I) = ModeCount(I)
        Coefficient(I) = Exp((a2(I) * d0(I) - a1(I) * d1(I)) / (a0(I) * a2(I) - a1(I) * a1(I)))
        Exponent(I) = (a0(I) * d1(I) - a1(I) * d0(I)) / (a0(I) * a2(I) - a1(I) * a1(I))

        'Set mode characteristics
        Graph1.ThisSet = I
        Select Case I
            Case 1
                Graph1.LegendText = "ORG"
                Graph1.SymbolData = G_CROSS_PLUS
                Graph1.ColorData = G_BLUE
            Case 2
                Graph1.LegendText = "SD"
                Graph1.SymbolData = G_CROSS_TIMES
                Graph1.ColorData = G_GREEN
            Case 3
                Graph1.LegendText = "E"
                Graph1.SymbolData = G_TRIANGLE_UP
                Graph1.ColorData = G_RED
        End Select

    Next I
    Screen.MousePointer = Default
End Sub

Sub LoadSampleData (XLower As Single, XUpper As Single, Normalize As Integer)
    Dim Mode        As Integer
    Const NumModes = 1
    Graph1.NumSets = NumModes
    Dim I As Integer

    'Array subscripting
    '       1 = Sample Mode

    ReDim ModeCount(NumModes)   As Integer

    Dim PI                      As Double
    ReDim Coefficient(NumModes) As Single
    ReDim Exponent(NumModes)    As Single

    Dim SampleHandle            As Integer
    Dim RecCount                As Integer

    ReDim TempLog(NumModes)     As Double
    ReDim TempDiv(NumModes)     As Double
    ReDim a0(NumModes)          As Integer
    ReDim a1(NumModes)          As Double
    ReDim a2(NumModes)          As Double
    ReDim d0(NumModes)          As Double
    ReDim d1(NumModes)          As Double

    For I = 1 To NumModes
        ModeCount(I) = 0
        a1(I) = 0#
        a2(I) = 0#
```

```
    d0(I) = 0#
    d1(I) = 0#
Next I
RecCount = 0

SampleHandle = DBHandle
more = AgiViewFirst(SampleHandle)
Screen.MousePointer = HourGlass
Do While more
    AgiViewGet SampleHandle, SampleRDS, SampleRec
    RecCount = RecCount + 1

    'Set appropriate Mode and update correct ModeCount
    Mode = 1
    ModeCount(Mode) = ModeCount(Mode) + 1

    'Calcualte the Muliplier, PI
    PI = SampleRec.PI
    'PI = SampleRec.RELY * SampleRec.DBSize * SampleRec.CPLX * SampleRec.EXTIME
    'PI = PI * SampleRec.STOR * SampleRec.VIRT * SampleRec.TURN * SampleRec.ACAP
    'PI = PI * SampleRec.AEXP * SampleRec.PCAP * SampleRec.VEXP * SampleRec.LEXP
    'PI = PI * SampleRec.MODP * SampleRec.TOOL * SampleRec.SCED * SampleRec.RVOL

    'Calculate sums for Coefficient and Exponent
    TempLog(Mode) = Log(SampleRec.ACTKDSI)
    TempDiv(Mode) = Log(SampleRec.ACTEFFORT / PI)
    a1(Mode) = a1(Mode) + TempLog(Mode)
    a2(Mode) = a2(Mode) + TempLog(Mode) * TempLog(Mode)
    d0(Mode) = d0(Mode) + TempDiv(Mode)
    d1(Mode) = d1(Mode) + TempDiv(Mode) * TempLog(Mode)

    If SampleRec.ACTKDSI >= XLower And SampleRec.ACTKDSI <= XUpper Then
        'Load Data to the Graph
        Graph1.ThisSet = Mode
        Graph1.ThisPoint = ModeCount(Mode)
        If Normalize Then
            Graph1.GraphData = SampleRec.ACTEFFORT / PI
        Else
            Graph1.GraphData = SampleRec.ACTEFFORT
        End If
        Graph1.XPosData = SampleRec.ACTKDSI

        'Check for maximum Effort and update if needed
        If YMax < SampleRec.ACTEFFORT Then
            YMax = SampleRec.ACTEFFORT
        End If

    End If

    more = AgiViewNext(SampleHandle)
Loop

For I = 1 To NumModes
    a0(I) = ModeCount(I)
    Coefficient(I) = Exp((a2(I) * d0(I) - a1(I) * d1(I)) / (a0(I) * a2(I) - a1(I) * a1(I)))
    Exponent(I) = (a0(I) * d1(I) - a1(I) * d0(I)) / (a0(I) * a2(I) - a1(I) * a1(I))

    'Set mode characteristics
    Graph1.ThisSet = I
    Select Case I
        Case 1
            Graph1.LegendText = "SMP1"
            Graph1.SymbolData = G_CROSS_PLUS
            Graph1.ColorData = G_BLUE
        Case 2
            Graph1.LegendText = "SD"
            Graph1.SymbolData = G_CROSS_TIMES
            Graph1.ColorData = G_GREEN
        Case 3
            Graph1.LegendText = "E"
            Graph1.SymbolData = G_TRIANGLE_UP
            Graph1.ColorData = G_RED
    End Select
Next I
```

```
    Screen.MousePointer = Default
End Sub
```

Code for BoehmCal Form

```
Sub cmdCalibrate_Click ()
    Dim Mode        As String
    Dim ModeCount   As Integer
    Dim RecCount    As Integer

    Dim PI          As Double
    Dim CoeffOnly   As Single
    Dim ExpOnly     As Single
    Dim Coefficient As Single
    Dim Exponent    As Single

    Dim BoehmHandle As Single

    Dim TempQ       As Double
    Dim SumMMQ      As Double
    Dim SumQ2       As Double
    Dim TempLog     As Double
    Dim TempDiv     As Double
    Dim a0          As Integer
    Dim a1          As Double
    Dim a2          As Double
    Dim d0          As Double
    Dim d1          As Double

    If optOrganic.Value = True Then
        Mode = "ORG"
        ExpOnly = 1.05
    ElseIf optSemiD.Value = True Then
        Mode = "SD"
        ExpOnly = 1.12
    ElseIf optEmbedded.Value = True Then
        Mode = "E"
        ExpOnly = 1.2
    ElseIf optAllData.Value = True Then
        Mode = "ALL"
        ExpOnly = 1.16
    Else
        Mode = "UNK"
        ExpOnly = 1.16
    End If

    BoehmHandle = DBHandle
    more = AgiViewFirst(BoehmHandle)

    SumMMQ = 0#
    SumQ2 = 0#
    ModeCount = 0
    txtModeTotal.Text = Format$(ModeCount, "###0")
    RecCount = 0
    a1 = 0#
    a2 = 0#
    d0 = 0#
    d1 = 0#

    Do While more
        RecCount = RecCount + 1
        txtRecTotal.Text = Format$(RecCount, "###0")
        AgiViewGet BoehmHandle, BoehmRDS, BoehmRec
        If Trim(BoehmRec.Mode) = Trim(Mode) Or Mode = "ALL" Then
            ModeCount = ModeCount + 1
            txtModeTotal.Text = Format$(ModeCount, "###0")
            PI = BoehmRec.RELY * BoehmRec.DBSIZE * BoehmRec.CPLX * BoehmRec.EXTIME
            PI = PI * BoehmRec.STOR * BoehmRec.VIRT * BoehmRec.TURN * BoehmRec.ACAP
            PI = PI * BoehmRec.AEXP * BoehmRec.PCAP * BoehmRec.VEXP * BoehmRec.LEXP
            PI = PI * BoehmRec.MODP * BoehmRec.TOOL * BoehmRec.SCED * BoehmRec.RVOL

            'Calculate sums for Coefficient only
            TempQ = PI * (BoehmRec.ADJKDSI ^ ExpOnly)
            SumMMQ = SumMMQ + BoehmRec.ACTEFFORT * TempQ
            SumQ2 = SumQ2 + TempQ * TempQ

            'Calculate sums for Coefficient and Exponent
```

```
            TempLog = Log(BoehmRec.ADJKDSI)
            TempDiv = Log(BoehmRec.ACTEFFORT / PI)
            a1 = a1 + TempLog
            a2 = a2 + TempLog * TempLog
            d0 = d0 + TempDiv
            d1 = d1 + TempDiv * TempLog

        End If
        more = AgiViewNext(BoehmHandle)
    Loop

    If ModeCount > 0 Then
        CoeffOnly = SumMMQ / SumQ2
        a0 = ModeCount
        Coefficient = Exp((a2 * d0 - a1 * d1) / (a0 * a2 - a1 * a1))
        Exponent = (a0 * d1 - a1 * d0) / (a0 * a2 - a1 * a1)
    End If

    txtCoeffOnly.Text = Format$(CoeffOnly, "0.0000")
    txtCoefficient.Text = Format$(Coefficient, "0.0000")
    txtExponent.Text = Format$(Exponent, "0.0000")
End Sub

Sub cmdCancel_Click ()
    Unload FrmCalBm
End Sub

Sub Form_Load ()
    txtDBName.Text = DBFileSpec
End Sub
```

Code for BoehmDat Form

```
Sub cmdCancel_Click ()
    Text1.Text = ""
    Text2.Text = ""
    Unload FrmBoehm
End Sub

Sub cmdFirst_Click ()
    more = AgiViewFirst(DBHandle)
    ComputeEffort
End Sub

Sub cmdNext_Click ()
    more = AgiViewNext(DBHandle)
    ComputeEffort
End Sub

Sub cmdPrevious_Click ()
    more = AgiViewPrevious(DBHandle)
    ComputeEffort
End Sub

Sub ComputeEffort ()
    Dim Multiplier As Double

    AgiViewGet DBHandle, BoehmRDS, BoehmRec
    Field1.Text = BoehmRec.RELY
    Field2.Text = BoehmRec.DBSIZE
    Field3.Text = BoehmRec.CPLX
    Field4.Text = BoehmRec.EXTIME
    Field5.Text = BoehmRec.STOR
    Field6.Text = BoehmRec.VIRT
    Field7.Text = BoehmRec.TURN
    Field8.Text = BoehmRec.ACAP
    Field9.Text = BoehmRec.AEXP
    Field10.Text = BoehmRec.PCAP
    Field11.Text = BoehmRec.VEXP
    Field12.Text = BoehmRec.LEXP
    Field13.Text = BoehmRec.MODP
    Field14.Text = BoehmRec.TOOL
    Field15.Text = BoehmRec.SCED
    Field25.Text = BoehmRec.RVOL

    Field16.Text = BoehmRec.Num
    Field17.Text = BoehmRec.Year
    Field18.Text = BoehmRec.LANGUAGE
    Field19.Text = BoehmRec.Mode
    Field23.Text = BoehmRec.ESTEFFORT

    Field20.Text = BoehmRec.PI
    Field21.Text = BoehmRec.TOTKDSI
    Field22.Text = BoehmRec.ADJKDSI
    Field24.Text = BoehmRec.ACTEFFORT

    Multiplier = Val(Field1.Text) * Val(Field2.Text) * Val(Field3.Text) * Val(Field4.Text) *
Val(Field5.Text) * Val(Field6.Text) * Val(Field7.Text) * Val(Field8.Text)
    Multiplier = Multiplier * Val(Field9.Text) * Val(Field10.Text) * Val(Field11.Text) *
Val(Field12.Text) * Val(Field13.Text) * Val(Field14.Text) * Val(Field15.Text) * Val(Field25.Text)
    Text1.Text = Format$(Multiplier#, "0.#0")
    Select Case Trim(Field19.Text)
        Case "ORG"
            Effort# = 3.2 * Val(Field22.Text) ^ 1.05 * Multiplier#
        Case "SD"
            Effort# = 3# * Val(Field22.Text) ^ 1.12 * Multiplier#
        Case "E"
            Effort# = 2.8 * Val(Field22.Text) ^ 1.2 * Multiplier#
        Case Else
            'MsgBox "Not Reading Mode from Database."
            'Unload FrmBoehm
    End Select
    Text2.Text = Format$(Effort#, "#,###,##0.#")
End Sub
```

```
Sub Form_Load ()
    more = AgiViewFirst(DBHandle)
    ComputeEffort
End Sub
```

Code for BoehmTbl Form

```
Option Explicit

Sub cmdCancel_Click ()
    Unload frmBoehmTable
End Sub
```

Code for CompCret

```
Option Explicit
Dim ThesisFileSpec As String

Sub cmdCancel_Click ()
    Unload frmCompCreate 'Close ThesisHandle is in Unload
End Sub

Sub cmdClear_Click ()
    While AgiViewFirst(ThesisHandle)
        AgiViewDelete ThesisHandle
        If AgiError() <> 0 Then
            MsgBox "Record deletion error: " + AgiErrorText()
        End If
    Wend
    cmdCreate.Enabled = True
    cmdClear.Enabled = False
    cmdSave.Enabled = False
End Sub

Sub cmdCreate_Click ()
    Dim NumRec          As Integer
    Dim NumCheck        As Integer

    Randomize           'Reseeds random number generator

    cmdClear_Click      'Clear existing records if any

    Screen.MousePointer = HourGlass
    NumCheck = 0
    If chkCID.Value = True Then
        ThesisRec.Category = "CID"
        NumCheck = NumCheck + 1
        For NumRec = 1 To Int(Val(txtBCNum.Text))
            MakeThesisRecord Val(txtCIDCoeff.Text), Val(txtCIDExp.Text), Val(txtCIDSD.Text)
            If AgiError() <> 0 Then
                MsgBox "Could not create record for Comm/Identification."
                cmdClear_Click
                Screen.MousePointer = Default
                Exit Sub
            End If
        Next NumRec
    End If
    If chkNAV.Value = True Then
        ThesisRec.Category = "NAV"
        NumCheck = NumCheck + 1
        For NumRec = 1 To Int(Val(txtBCNum.Text))
            MakeThesisRecord Val(txtNAVCoeff.Text), Val(txtNAVExp.Text), Val(txtNAVSD.Text)
            If AgiError() <> 0 Then
                MsgBox "Could not create record for Navigation Sensors."
                cmdClear_Click
                Screen.MousePointer = Default
                Exit Sub
            End If
        Next NumRec
    End If
    If chkCAV.Value = True Then
        ThesisRec.Category = "CAV"
        NumCheck = NumCheck + 1
        For NumRec = 1 To Int(Val(txtBCNum.Text))
            MakeThesisRecord Val(txtCAVCoeff.Text), Val(txtCAVExp.Text), Val(txtCAVSD.Text)
            If AgiError() <> 0 Then
                MsgBox "Could not create record for Core Avionics."
                cmdClear_Click
                Screen.MousePointer = Default
                Exit Sub
            End If
        Next NumRec
    End If
    If chkECS.Value = True Then
        ThesisRec.Category = "ECS"
        NumCheck = NumCheck + 1
        For NumRec = 1 To Int(Val(txtBCNum.Text))
```

```
            MakeThesisRecord Val(txtECSCoeff.Text), Val(txtECSExp.Text), Val(txtECSSD.Text)
            If AgiError() <> 0 Then
                MsgBox "Could not create record for Electronic Combat."
                cmdClear_Click
                Screen.MousePointer = Default
                Exit Sub
            End If
        Next NumRec
    End If
    If chkOFF.Value = True Then
        ThesisRec.Category = "OFF"
        NumCheck = NumCheck + 1
        For NumRec = 1 To Int(Val(txtBCNum.Text))
            MakeThesisRecord Val(txtOFFCoeff.Text), Val(txtOFFExp.Text), Val(txtOFFSD.Text)
            If AgiError() <> 0 Then
                MsgBox "Could not create record for Offesive Sensors."
                cmdClear_Click
                Screen.MousePointer = Default
                Exit Sub
            End If
        Next NumRec
    End If
    If NumCheck = 0 Then
        ErrMsg = "Please select at least one category from the check box list."
        MsgBox ErrMsg
        Screen.MousePointer = Default
        Exit Sub
    End If

    cmdCreate.Enabled = False
    cmdClear.Enabled = True
    cmdSave.Enabled = True
    Screen.MousePointer = Default
End Sub

Sub cmdSave_Click ()
    Dim SaveHandle As Integer
    Dim Filetitle As String

    'Get new FileSpec
    On Error Resume Next
    CMDialog1.DefaultExt = "AGI"
    CMDialog1.Flags = OFN_PATHMUSTEXIST + OFN_OVERWRITEPROMPT + OFN_HIDEREADONLY +
OFN_EXTENSIONDIFFERENT
    CMDialog1.Action = DLG_FILE_SAVE
    If Err = 32755 Then         'user selcted cancel button
        Exit Sub
    End If
    On Error GoTo 0

    'Test to see if new file is already open or exists
    Screen.MousePointer = HourGlass
    If CMDialog1.Filetitle = frmMain!CMDialog1.Filetitle Then
        AgiViewClose DBHandle               'Close Open Copy
        Kill CMDialog1.Filename             'Delete Previous open copy
        frmMain!mnuCloseDB.Enabled = False  'Reset menu selection
        frmMain!mnuViewGraph.Enabled = False
        frmMain!mnuViewTable.Enabled = False
        frmMain!mnuViewData.Visible = False
        frmMain!mnuCalBoehm.Enabled = False
        frmMain!mnuStatEval.Enabled = False
    ElseIf CMDialog1.Filetitle = "COMPTHS.AGI" Then
        Exit Sub         'File update will be automatic upon Cancel_Click
    ElseIf Dir$(CMDialog1.Filename) <> "" Then     'If File exists
        Kill CMDialog1.Filename             'Delete unopened coy
    End If

    ' Open blank file to copy into
    SaveHandle = AgiFreeFile()
    Filetitle = CMDialog1.Filetitle
    AgiViewOpen SaveHandle, Filetitle, "CAUR"
    If AgiError() <> 0 Then
        MsgBox "File Creation Error: " + AgiErrorText()
    End If
```

```
    ' Copy current file into blank copy for saving
    more = AgiViewFirst(ThesisHandle)
    While more
        AgiViewGet ThesisHandle, ThesisRDS, ThesisRec
        AgiViewAdd SaveHandle, ThesisRDS, ThesisRec
        If AgiError() <> 0 Then
            MsgBox "Error adding record to save file." + NL() + "File results undetermined."
            AgiViewClose (SaveHandle)
            Screen.MousePointer = Default
            Exit Sub
        End If
        more = AgiViewNext(ThesisHandle)
    Wend
    AgiViewClose (SaveHandle)       'ThesisHandle remains open
    Screen.MousePointer = Default
End Sub

Function FindEDSI (ADDKDSI As Double, MODKDSI As Double, DELKDSI As Double) As Double
    'Solution is found by weighting the KDSI values
    'These weightings are from the Softcost-R Manual page R-83
    'Note the wieghtings for lines and modules have been added so as to include
    'the effect of altering both the line and the module
    FindEDSI = .53 * ADDKDSI + (.27 + .24) * MODKDSI + (.15 + .11) * DELKDSI
End Function

Function FindEXTIME (PerTime As Double) As Double
    If PerTime <= 65# Then
        FindEXTIME = 1#
    Else
        FindEXTIME = 1.82 * (PerTime / 100#) ^ 1.305
    End If
End Function

Function FindLearn (BCNUM As Integer) As Double
    'This function assumes the realtionship holds true for 6 block changes.
    'But SYSCON states relationship holds for 6 years.
    'Future may need to adjust for years if block changes are not 1 year
    'The exponent comes from SYSCON, Thesis TAble 5
    If BCNUM >= 2 And BCNUM <= 6 Then
        'No coefficient => normalize to 1.0
        'Note Effect is ratio to previous BC and not the first BC
        FindLearn = (BCNUM ^ (-.375)) / ((BCNUM - 1) ^ (-.375))
    Else
        FindLearn = 1#
    End If
End Function

Function FindSTOR (PerMem As Double) As Double
    If PerMem <= 65# Then
        FindSTOR = 1#
    Else
        FindSTOR = 1.94 * (PerMem / 100#) ^ 1.425
    End If
End Function

Sub Form_Load ()
    Dim Filetitle As String
    Dim FileSpec As String

    'Open new file (agi formats only).
    ThesisHandle = AgiFreeFile()
    Filetitle = "COMPTHS.AGI"
    AgiViewOpen ThesisHandle, Filetitle, "CAUR"
    If AgiError() <> 0 Then
        MsgBox "File Open Error: " + AgiErrorText()
    End If
    Table1.ViewHandle = ThesisHandle

    'Enable Appropriate Buttons
    more = AgiViewFirst(ThesisHandle)
    If more Then
        cmdCreate.Enabled = False
    Else
        cmdClear.Enabled = False
```

```
        cmdSave.Enabled = False
    End If

    'Set FileSpec for Save Dialog Box
    If Right(File1.Path, 1) <> "\" Then
        FileSpec = File1.Path + "\" + Filetitle
    Else
        FileSpec = File1.Path + Filetitle
    End If
    CMDialog1.Filename = FileSpec
End Sub

Sub Form_Unload (Cancel As Integer)
    AgiViewClose ThesisHandle
End Sub

Sub MakeThesisRecord (Coefficient As Double, Exponent As Double, StdDev As Double)
    Dim NumBC           As Integer
    Dim EffSDPercent    As Single
    Dim SD              As Single
    Dim EAF             As Double
    Dim EstEffort       As Single
    Dim LearnEffect     As Double

    SD = Val(txtFactorSD.Text)

    ThesisRec.BCNUM = Int((Val(txtBCNum.Text)) * Rnd + 1)

    ThesisRec.RELY = NDEV(1, SD)
    ThesisRec.DBSIZE = NDEV(1, SD)
    ThesisRec.CPLX = NDEV(1, SD)
    ThesisRec.TIMEUTIL = 30# + 70# * Rnd
    ThesisRec.EXTIME = FindEXTIME(Val(ThesisRec.TIMEUTIL))
    ThesisRec.MEMUTIL = 30# + 70# * Rnd
    ThesisRec.STOR = FindSTOR(Val(ThesisRec.MEMUTIL))
    ThesisRec.VIRT = NDEV(1, SD)
    ThesisRec.TURN = NDEV(1, SD)
    ThesisRec.ACAP = NDEV(1, SD)
    LearnEffect = FindLearn(Int(ThesisRec.BCNUM))
    'Spread Learning Effect to AEXP and LEXP
    ThesisRec.AEXP = NDEV(1, SD) * Sqr(LearnEffect)
    ThesisRec.PCAP = NDEV(1, SD)
    ThesisRec.VEXP = NDEV(1, SD)
    ThesisRec.LEXP = NDEV(1, SD) * Sqr(LearnEffect)
    ThesisRec.MODP = NDEV(1, SD)
    ThesisRec.TOOL = NDEV(1, SD)
    ThesisRec.SCED = NDEV(1, SD)
    ThesisRec.RVOL = NDEV(1, SD)
    EAF = ThesisRec.RELY * ThesisRec.DBSIZE * ThesisRec.CPLX * ThesisRec.EXTIME
    EAF = EAF * ThesisRec.STOR * ThesisRec.VIRT * ThesisRec.TURN * ThesisRec.ACAP
    EAF = EAF * ThesisRec.AEXP * ThesisRec.PCAP * ThesisRec.VEXP * ThesisRec.LEXP
    EAF = EAF * ThesisRec.MODP * ThesisRec.TOOL * ThesisRec.SCED * ThesisRec.RVOL
    ThesisRec.PI = EAF
    ThesisRec.ENTROPY = 1#
    ThesisRec.ACTKDSI = Val(txtDevKDSI.Text) * Rnd
    ThesisRec.ADDKDSI = ThesisRec.ACTKDSI * (Val(txtPerAdd.Text) / 100) * NDEV(1,
Val(txtPerSD.Text) / 100)
    ThesisRec.MODKDSI = ThesisRec.ACTKDSI * (Val(txtPerMod.Text) / 100) * NDEV(1,
Val(txtPerSD.Text) / 100)
    ThesisRec.DELKDSI = ThesisRec.ACTKDSI * (Val(txtPerDel.Text) / 100) * NDEV(1,
Val(txtPerSD.Text) / 100)
    ThesisRec.EDSI = FindEDSI(Val(ThesisRec.ADDKDSI), Val(ThesisRec.MODKDSI),
Val(ThesisRec.DELKDSI))
    EstEffort = EAF * Coefficient * (ThesisRec.ACTKDSI ^ Exponent) * (ThesisRec.EDSI /
ThesisRec.ACTKDSI)
    'EffSDPercent is the % of SD at the EDSI value
    EffSDPercent = StdDev / ThesisRec.EDSI
    ThesisRec.ACTEFFORT = Exp(Log(EstEffort) + NDEV(0, Log(1 + EffSDPercent)))
    AgiViewAdd ThesisHandle, ThesisRDS, ThesisRec

End Sub

Sub txtBCNum_LostFocus ()
    Dim BCNUM As Integer
```

```
    BCNUM = Int(Val(txtBCNum.Text))
    If BCNUM > 0 Then
        txtBCNum.Text = Str(BCNUM)
    Else
        ErrMsg = "Please select a positve value for" + NL()
        ErrMsg = ErrMsg + "the number of records."
        MsgBox ErrMsg
        txtBCNum.Text = "10"
        txtBCNum.SetFocus
    End If
End Sub
```

Code for CompGrph Form

```
Option Explicit
Option Base 1

Sub cmdCancel_Click ()
    Graph1.DataReset = G_All_Data
    Unload frmCompGraph
End Sub

Sub cmdNorm_Click ()
    If NormState = Norm_Off Then
        cmdNorm.Caption = "Normalize Off"
        NormState = Norm_On
    Else
        cmdNorm.Caption = "Normalize On"
        NormState = Norm_Off
    End If

    Select Case DBType
        Case BoehmDB
        Case SampleDB
        Case ThesisDB
        Case CompThesisDB
            LoadCompData XMin, XMax, NormState
        Case UnknownDB
    End Select

    Graph1.DrawMode = G_Draw
    If FileGraph = 1 Then
        Graph1.DrawMode = G_Write
    End If
End Sub

Sub cmdZoom_Click ()
    Dim TempYMax As Single
    Dim TempYMin As Single
    Dim TempXMax As Single
    Dim TempXMin As Single
    Dim Redraw As Integer

    Redraw = False

    TempYMax = YMax
    TempYMin = YMin
    TempXMax = XMax
    TempXMin = XMin

    frmZoomData.Show Modal

    If XMax <> TempXMax Or XMin <> TempXMin Then
        Redraw = True
        Graph1.DataReset = G_All_Data
        Select Case DBType
            Case BoehmDB
            Case SampleDB
            Case ThesisDB
            Case CompThesisDB
                LoadCompData XMin, XMax, NormState
            Case UnknownDB
        End Select
    End If

    If YMax <> TempYMax Then
        Redraw = True
        Graph1.YAxisMax = YMax
    End If

    If YMin <> TempYMin Then
        Redraw = True
        Graph1.YAxisMin = YMin
    End If

    If Redraw Then
```

```
        Graph1.DrawMode = G_Draw
    End If
    If FileGraph = 1 Then
        Graph1.DrawMode = G_Write
    End If
End Sub

Sub Form_Load ()
    txtDBName.Text = DBFileSpec
    Screen.MousePointer = Hourglass

    'set filename for saving graph
    Graph1.ImageFile = "compgrph"

    Graph1.NumPoints = 100
    Graph1.IndexStyle = 1   'Enhanced index style permits access to
                            'graph 2-D arrays for scatter data.

    Graph1.YAxisStyle = 2   'Permits user defined Y-origin through YAxisMin,
    Graph1.YAxisTicks = 8

    'Set default Axis limits for graph
    YMax = 1#       'Will increase upon reading database
    YMin = 0#
    XMax = 3000#    'May decrease to fit x range of data
    XMin = 0#

    WindowState = Maximized
    NormState = Norm_Off
    Graph1.LeftTitle = "MM"
    Graph1.BottomTitle = "Size in EDSI"
    Graph1.DrawMode = G_Draw

    Select Case DBType
        Case BoehmDB
        Case SampleDB
        Case ThesisDB, CompThesisDB
            LoadCompData XMin, XMax, NormState
        Case UnknownDB
    End Select
    Screen.MousePointer = Hourglass

    'Set Y-axis scale
    Graph1.YAxisMax = YMax
    Graph1.YAxisMin = YMin
    Graph1.DrawMode = G_Draw
    If FileGraph = 1 Then
        Graph1.DrawMode = G_Write
    End If
    Screen.MousePointer = Default
End Sub

Sub LoadCompData (XLower As Single, XUpper As Single, Normalize As Integer)
    Screen.MousePointer = Hourglass
    Dim Mode     As Integer
    Dim CIDMode  As Integer
    Dim NAVMode  As Integer
    Dim CAVMode  As Integer
    Dim ECSMode  As Integer
    Dim OFFMode  As Integer

    Dim NumModes As Integer
    Const CatPresent = 1
    Const CatAbsent = 0
    Dim CIDExist As Integer
    Dim NAVExist As Integer
    Dim CAVExist As Integer
    Dim ECSExist As Integer
    Dim OFFExist As Integer

    Dim I As Integer

    'Array subscripting
    '       1 = Comm / ID
```

```
'       2 = Navigation Sensors
'       3 = Core Avionics
'       4 = Electronic Combat Systems
'       5 = Offensive Sensors

Dim PI                      As Double
Dim CompHandle              As Integer
Dim RecCount                As Integer

CompHandle = DBHandle
more = AgiViewFirst(CompHandle)
CIDExist = CatAbsent
NAVExist = CatAbsent
CAVExist = CatAbsent
ECSExist = CatAbsent
OFFExist = CatAbsent
Do While more
    AgiViewGet CompHandle, ThesisRDS, ThesisRec
    Select Case Trim(ThesisRec.Category)
        Case "CID"
            CIDExist = CatPresent
        Case "NAV"
            NAVExist = CatPresent
        Case "CAV"
            CAVExist = CatPresent
        Case "ECS"
            ECSExist = CatPresent
        Case "OFF"
            OFFExist = CatPresent
    End Select
    more = AgiViewNext(CompHandle)
Loop

NumModes = 0
If CIDExist Then
    NumModes = NumModes + 1
    CIDMode = NumModes
End If
If NAVExist Then
    NumModes = NumModes + 1
    NAVMode = NumModes
End If
If CAVExist Then
    NumModes = NumModes + 1
    CAVMode = NumModes
End If
If ECSExist Then
    NumModes = NumModes + 1
    ECSMode = NumModes
End If
If OFFExist Then
    NumModes = NumModes + 1
    OFFMode = NumModes
End If

'Intialize all mode arrays to zero
ReDim ModeCount(NumModes)   As Integer
Graph1.NumSets = NumModes
For I = 1 To NumModes
    ModeCount(I) = 0
Next I

RecCount = 0
more = AgiViewFirst(CompHandle)
Do While more
    AgiViewGet CompHandle, ThesisRDS, ThesisRec

    RecCount = RecCount + 1

    'Set appropriate Mode and update correct ModeCount
    Select Case Trim(ThesisRec.Category)
        Case "CID"
            Mode = CIDMode
        Case "NAV"
```

```
                Mode = NAVMode
            Case "CAV"
                Mode = CAVMode
            Case "ECS"
                Mode = ECSMode
            Case "OFF"
                Mode = OFFMode
            Case Else
                MsgBox "Unrecognized Mode at record " + Str(RecCount) + "."
                Mode = 1
        End Select
        ModeCount(Mode) = ModeCount(Mode) + 1

        If ThesisRec.EDSI >= XLower And ThesisRec.EDSI <= XUpper Then
            'Load Data to the Graph
            Graph1.ThisSet = Mode
            Graph1.ThisPoint = ModeCount(Mode)
            If Normalize Then
                Graph1.GraphData = ThesisRec.ACTEFFORT / ThesisRec.PI
            Else
                Graph1.GraphData = ThesisRec.ACTEFFORT
            End If
            Graph1.XPosData = ThesisRec.EDSI

            'Check for maximum Effort and update if needed
            If YMax < ThesisRec.ACTEFFORT Then
                YMax = ThesisRec.ACTEFFORT
            End If
        End If

        more = AgiViewNext(CompHandle)
    Loop

    'Set mode characteristics
    If CIDExist Then
        Graph1.ThisSet = CIDMode
        Graph1.LegendText = "C/ID"
        Graph1.SymbolData = G_CROSS_PLUS
        Graph1.ColorData = G_BLUE
    End If
    If NAVExist Then
        Graph1.ThisSet = NAVMode
        Graph1.LegendText = "NAV"
        Graph1.SymbolData = G_CROSS_TIMES
        Graph1.ColorData = G_GREEN
    End If
    If CAVExist Then
        Graph1.ThisSet = CAVMode
        Graph1.LegendText = "CORE"
        Graph1.SymbolData = G_TRIANGLE_UP
        Graph1.ColorData = G_RED
    End If
    If ECSExist Then
        Graph1.ThisSet = ECSMode
        Graph1.LegendText = "E C"
        Graph1.SymbolData = G_DIAMOND
        Graph1.ColorData = G_BLACK
    End If
    If OFFExist Then
        Graph1.ThisSet = OFFMode
        Graph1.LegendText = "OFF"
        Graph1.SymbolData = G_SQUARE
        Graph1.ColorData = G_BROWN
    End If
    Screen.MousePointer = Default
End Sub
```

Code for DataConv Form

```
Option Explicit

Sub cmdCancel_Click ()
    Unload frmDataConv
End Sub

Sub cmdConvert_Click ()
    Dim SaveHandle As Integer
    Dim Filetitle As String
    Dim SourceSpec As String
    Dim DestSpec As String
    Dim UpToExt As Integer

    'Filetitle = "GOOD.AGI"
    Filetitle = "SHELL.DBF"
    If Right(File1.Path, 1) <> "\" Then
        SourceSpec = File1.Path + "\" + Filetitle
    Else
        SourceSpec = File1.Path + Filetitle
    End If


    'Get new FileSpec
    UpToExt = InStr(frmMain!CMDialog1.Filename, ".")
    DestSpec = Left(frmMain!CMDialog1.Filename, UpToExt) + "DBF"
    MsgBox "Destination filespec is " + DestSpec
    CMDialog1.Filename = DestSpec
    'CMDialog1.DefaultExt = "AGI"
    CMDialog1.DefaultExt = "DBF"
    CMDialog1.Flags = OFN_PATHMUSTEXIST + OFN_OVERWRITEPROMPT + OFN_HIDEREADONLY +
OFN_EXTENSIONDIFFERENT
    On Error Resume Next
    CMDialog1.Action = DLG_FILE_SAVE
    If Err = 32755 Then         'user selcted cancel button
        Exit Sub
    End If
    On Error GoTo 0

    MsgBox "Save Filetitle is " + CMDialog1.Filetitle + "."
    'Test to see if new file is already open or exists
    Screen.MousePointer = HourGlass
    If CMDialog1.Filetitle = frmMain!CMDialog1.Filetitle Then
        'AgiViewClose DBHandle               'Close Open Copy
        'Kill CMDialog1.Filename             'Delete Previous open copy
        'frmMain!mnuCloseDB.Enabled = False  'Reset menu selection
        'frmMain!mnuViewGraph.Enabled = False
        'frmMain!mnuViewTable.Enabled = False
        'frmMain!mnuViewData.Visible = False
        'frmMain!mnuCalBoehm.Enabled = False
        'frmMain!mnuStatEval.Enabled = False
    'ElseIf CMDialog1.Filetitle = "THSFILE.AGI" Then
        'Exit Sub         'File update will be automatic upon Cancel_Click
    ElseIf Dir$(CMDialog1.Filename) <> "" Then     'If File exists
        Kill CMDialog1.Filename             'Delete unopened coy
    End If

    'Copy shell.dbf file to hold new input
    MsgBox "SourceSpec is " + SourceSpec
    MsgBox "DestSpec is " + CMDialog1.Filename
    On Error Resume Next
    FileCopy SourceSpec, CMDialog1.Filename
    If Err = 55 Then  'File already open
        ErrMsg = "Cannot copy an open file.  Close and try again."
        MsgBox ErrMsg
        Screen.MousePointer = Default
        Exit Sub
    End If
    On Error GoTo 0

    ' Open blank file in dbase format to copy into
    SaveHandle = AgiFreeFile()
    Filetitle = CMDialog1.Filetitle
```

```
    AgiViewOpen SaveHandle, Filetitle, "CAUR"
    If AgiError() <> 0 Then
        MsgBox "File Creation Error: " + AgiErrorText()
    End If

    ' Copy current file into blank copy for saving
    ' Current DBHandle must refer to a thesisDB type
    more = AgiViewFirst(DBHandle)
    While more
        AgiViewGet DBHandle, ThesisRDS, ThesisRec
        'ThesisRec.BCNUM = OldThesisRec.BCNUM
        'ThesisRec.CATEGORY = OldThesisRec.CATEGORY
        'ThesisRec.RELY = OldThesisRec.RELY
        'ThesisRec.DBSIZE = OldThesisRec.DBSIZE
        'ThesisRec.CPLX = OldThesisRec.CPLX
        'ThesisRec.TIMEUTIL = OldThesisRec.TIMEUTIL
        'ThesisRec.EXTIME = OldThesisRec.EXTIME
        'ThesisRec.MEMUTIL = OldThesisRec.MEMUTIL
        'ThesisRec.STOR = OldThesisRec.STOR
        'ThesisRec.VIRT = OldThesisRec.VIRT
        'ThesisRec.TURN = OldThesisRec.TURN
        'ThesisRec.ACAP = OldThesisRec.ACAP
        'ThesisRec.AEXP = OldThesisRec.AEXP
        'ThesisRec.PCAP = OldThesisRec.PCAP
        'ThesisRec.VEXP = OldThesisRec.VEXP
        'ThesisRec.LEXP = OldThesisRec.LEXP
        'ThesisRec.MODP = OldThesisRec.MODP
        'ThesisRec.TOOL = OldThesisRec.TOOL
        'ThesisRec.SCED = OldThesisRec.SCED
        'ThesisRec.RVOL = OldThesisRec.RVOL
        'ThesisRec.PI = OldThesisRec.PI
        'ThesisRec.ENTROPY = OldThesisRec.ENTROPY
        'ThesisRec.ACTKDSI = OldThesisRec.ACTKDSI
        'ThesisRec.ADDKDSI = OldThesisRec.ADDKDSI
        'ThesisRec.MODKDSI = OldThesisRec.MODKDSI
        'ThesisRec.DELKDSI = OldThesisRec.DELKDSI
        'ThesisRec.EDSI = OldThesisRec.ESDI
        'ThesisRec.ACTEFFORT = OldThesisRec.ACTEFFORT

        AgiViewAdd SaveHandle, ThesisRDS, ThesisRec
        If AgiError() <> 0 Then
            MsgBox "Error adding record to save file." + NL() + "File results undetermined."
            AgiViewClose (SaveHandle)
            Screen.MousePointer = Default
            Exit Sub
        End If
        more = AgiViewNext(DBHandle)
    Wend
    AgiViewClose (SaveHandle)       'DBHandle remains open
    Screen.MousePointer = Default
End Sub

Sub Form_Load ()
    Dim Filetitle As String
    Dim FileSpec As String

    'Open new file (agi formats only).
    'ThesisHandle = AgiFreeFile()
    'Filetitle = "SHELL.DBF"
    'AgiViewOpen ThesisHandle, Filetitle, "CAUR"
    'If AgiError() <> 0 Then
        'MsgBox "File Open Error: " + AgiErrorText()
    'End If
    'Table1.ViewHandle = ThesisHandle

    'Enable Appropriate Buttons
    'more = AgiViewFirst(ThesisHandle)
    'If more Then
        'cmdCreate.Enabled = False
    'Else
        'cmdClear.Enabled = False
        'cmdSave.Enabled = False
    'End If
```

```
    'Set FileSpec for Save Dialog Box
    'If Right(File1.Path, 1) <> "\" Then
        'FileSpec = File1.Path + "\" + Filetitle
    'Else
        'FileSpec = File1.Path + Filetitle
    'End If
    'CMDialog1.Filename = FileSpec
End Sub
```

```
Code for DBCreate Form

Option Explicit
Dim SampleFileSpec As String

Sub cmdCancel_Click ()
    'cmdClear_Click
    Unload frmDBCreate  'Close SampleHandle is in Unload
End Sub

Sub cmdClear_Click ()
    While AgiViewFirst(SampleHandle)
        AgiViewDelete SampleHandle
        If AgiError() <> 0 Then
            MsgBox "Record deletion error: " + AgiErrorText()
        End If
    Wend
    cmdCreate.Enabled = True
    cmdClear.Enabled = False
    cmdSave.Enabled = False
End Sub

Sub cmdCreate_Click ()
    Dim NumRec          As Integer
    Dim EffSDPercent    As Single
    Dim SD              As Single
    Dim EAF             As Double
    Dim EstEffort       As Single

    Randomize           'Reseeds random number generator
    SD = Val(txtFactorSD.Text)

    cmdClear_Click      'Clear existing records if any
    Screen.MousePointer = HourGlass
    For NumRec = 1 To Int(txtNumRec.Text)
        Debug.Print "Record number " + Str$(NumRec)
        SampleRec.Num = NumRec
        SampleRec.Type = "XXX"
        SampleRec.RELY = NDEV(1, SD)
        SampleRec.DBSIZE = NDEV(1, SD)
        SampleRec.CPLX = NDEV(1, SD)
        SampleRec.EXTIME = NDEV(1, SD)
        SampleRec.STOR = NDEV(1, SD)
        SampleRec.VIRT = NDEV(1, SD)
        SampleRec.TURN = NDEV(1, SD)
        SampleRec.ACAP = NDEV(1, SD)
        SampleRec.AEXP = NDEV(1, SD)
        SampleRec.PCAP = NDEV(1, SD)
        SampleRec.VEXP = NDEV(1, SD)
        SampleRec.LEXP = NDEV(1, SD)
        SampleRec.MODP = NDEV(1, SD)
        SampleRec.TOOL = NDEV(1, SD)
        SampleRec.SCED = NDEV(1, SD)
        SampleRec.RVOL = NDEV(1, SD)
        SampleRec.ACTKDSI = Int((Val(txtMaxKDSI.Text) - Val(txtMinKDSI.Text) + 1) * Rnd +
Val(txtMinKDSI.Text))
        EAF = SampleRec.RELY * SampleRec.DBSIZE * SampleRec.CPLX * SampleRec.EXTIME
        EAF = EAF * SampleRec.STOR * SampleRec.VIRT * SampleRec.TURN * SampleRec.ACAP
        EAF = EAF * SampleRec.AEXP * SampleRec.PCAP * SampleRec.VEXP * SampleRec.LEXP
        EAF = EAF * SampleRec.MODP * SampleRec.TOOL * SampleRec.SCED * SampleRec.RVOL
        SampleRec.PI = EAF
        EstEffort = EAF * Val(txtCoefficient.Text) * ((SampleRec.ACTKDSI) ^ Val(txtExponent.Text))
        'EffSDPercent is the % of SD at the midpoint from Min & Max KDSI
        EffSDPercent = Val(txtEffortSD.Text) / (Val(txtCoefficient.Text) * (((Val(txtMinKDSI.Text)
+ Val(txtMaxKDSI.Text)) / 2) ^ Val(txtExponent.Text)))
        SampleRec.ACTEFFORT = Exp(Log(EstEffort) + NDEV(0, Log(1 + EffSDPercent)))
        AgiViewAdd SampleHandle, SampleRDS, SampleRec
        If AgiError() <> 0 Then
            MsgBox "Could not add record " + Str(NumRec) + "."
            cmdClear_Click
            Exit Sub
        End If
    Next NumRec
    cmdCreate.Enabled = False
```

```
    cmdClear.Enabled = True
    cmdSave.Enabled = True
    Screen.MousePointer = Default
End Sub

Sub cmdSave_Click ()
    Dim SaveHandle As Integer
    Dim Filetitle As String

    'Get new FileSpec
    On Error Resume Next
    CMDialog1.DefaultExt = "AGI"
    CMDialog1.Flags = OFN_PATHMUSTEXIST + OFN_OVERWRITEPROMPT + OFN_HIDEREADONLY +
OFN_EXTENSIONDIFFERENT
    CMDialog1.Action = DLG_FILE_SAVE
    If Err = 32755 Then         'user selcted cancel button
        Exit Sub
    End If
    On Error GoTo 0

    Debug.Print "Save Filetitle is " + CMDialog1.Filetitle + "."
    'Test to see if new file is already open or exists
    Screen.MousePointer = HourGlass
    If CMDialog1.Filetitle = frmMain!CMDialog1.Filetitle Then
        AgiViewClose DBHandle               'Close Open Copy
        Kill CMDialog1.Filename             'Delete Previous open copy
        frmMain!mnuCloseDB.Enabled = False  'Reset menu selection
        frmMain!mnuViewGraph.Enabled = False
        frmMain!mnuViewTable.Enabled = False
        frmMain!mnuViewData.Visible = False
        frmMain!mnuCalBoehm.Enabled = False
        frmMain!mnuStatEval.Enabled = False
    ElseIf CMDialog1.Filetitle = "NEWFILE.AGI" Then
        Exit Sub         'File update will be automatic upon Cancel_Click
    ElseIf Dir$(CMDialog1.Filename) <> "" Then     'If File exists
        Kill CMDialog1.Filename             'Delete unopened coy
    End If

    ' Open blank file to copy into
    SaveHandle = AgiFreeFile()
    Filetitle = CMDialog1.Filetitle
    AgiViewOpen SaveHandle, Filetitle, "CAUR"
    If AgiError() <> 0 Then
        MsgBox "File Creation Error: " + AgiErrorText()
    End If

    ' Copy current file into blank copy for saving
    more = AgiViewFirst(SampleHandle)
    While more
        AgiViewGet SampleHandle, SampleRDS, SampleRec
        AgiViewAdd SaveHandle, SampleRDS, SampleRec
        If AgiError() <> 0 Then
            MsgBox "Error adding record to save file." + NL() + "File results undetermined."
            AgiViewClose (SaveHandle)
            Screen.MousePointer = Default
            Exit Sub
        End If
        more = AgiViewNext(SampleHandle)
    Wend
    AgiViewClose (SaveHandle)       'SampleHandle is remains open
    Screen.MousePointer = Default
End Sub

Sub Form_Load ()
    Dim Filetitle As String
    Dim FileSpec As String

    'Open new file (agi formats only).
    SampleHandle = AgiFreeFile()
    Filetitle = "NEWFILE.AGI"
    AgiViewOpen SampleHandle, Filetitle, "CAUR"
    If AgiError() <> 0 Then
        MsgBox "File Open Error: " + AgiErrorText()
    End If
```

```
    Table1.ViewHandle = SampleHandle

    'Enable Appropriate Buttons
    more = AgiViewFirst(SampleHandle)
    If more Then
        cmdCreate.Enabled = False
    Else
        cmdClear.Enabled = False
        cmdSave.Enabled = False
    End If

    'Set FileSpec for Save Dialog Box
    If Right(File1.Path, 1) <> "\" Then
        FileSpec = File1.Path + "\" + Filetitle
    Else
        FileSpec = File1.Path + Filetitle
    End If
    CMDialog1.Filename = FileSpec
End Sub

Sub Form_Unload (Cancel As Integer)
    AgiViewClose SampleHandle
End Sub

Sub txtEffortSD_LostFocus ()
    If Val(txtEffortSD.Text) < 0# Then
        MsgBox "The Effort Standard Deviation cannot be negative."
        txtEffortSD.Text = "30.0"
        txtEffortSD.SetFocus
        txtEffortSD.SelStart = 0
        txtEffortSD.SelLength = 64000
    End If
End Sub

Sub txtNumRec_LostFocus ()
    Dim NumRec As Integer
    NumRec = Int(Val(txtNumRec.Text))
    If NumRec > 0 Then
        txtNumRec.Text = Str(NumRec)
    Else
        ErrMsg = "Please select a positve value for" + NL()
        ErrMsg = ErrMsg + "the number of records."
        MsgBox ErrMsg
        txtNumRec.Text = "10"
        txtNumRec.SetFocus
    End If
End Sub
```

Code for GenTable Form

```
Option Explicit

Sub cmdCancel_Click ()
    Unload frmTable08
End Sub
```

```
Code for Mair hes Form

Option Explicit

Function FindDBType () As Integer

    Dim ValidBoehm     As Integer
    Dim ValidSample    As Integer
    Dim ValidThesis    As Integer
    Dim ValidOldThesis As Integer
    Dim ThesisCategory As String
    Dim ThesisBCNum    As Integer

    'put record pointer on first record
    more = AgiViewFirst(DBHandle)

    'Check Boehm's original database structure
    AgiViewGet DBHandle, BoehmRDS, BoehmRec
    If IsNull(BoehmRec.Num) Then
        ValidBoehm = False
    ElseIf IsNull(BoehmRec.Type) Then
        ValidBoehm = False
    ElseIf IsNull(BoehmRec.LANGUAGE) Then
        ValidBoehm = False
    ElseIf IsNull(BoehmRec.RELY) Then
        ValidBoehm = False
    ElseIf IsNull(BoehmRec.DBSIZE) Then
        ValidBoehm = False
    ElseIf IsNull(BoehmRec.CPLX) Then
        ValidBoehm = False
    ElseIf IsNull(BoehmRec.EXTIME) Then
        ValidBoehm = False
    ElseIf IsNull(BoehmRec.STOR) Then
        ValidBoehm = False
    ElseIf IsNull(BoehmRec.VIRT) Then
        ValidBoehm = False
    ElseIf IsNull(BoehmRec.TURN) Then
        ValidBoehm = False
    ElseIf IsNull(BoehmRec.ACAP) Then
        ValidBoehm = False
    ElseIf IsNull(BoehmRec.AEXP) Then
        ValidBoehm = False
    ElseIf IsNull(BoehmRec.PCAP) Then
        ValidBoehm = False
    ElseIf IsNull(BoehmRec.VEXP) Then
        ValidBoehm = False
    ElseIf IsNull(BoehmRec.LEXP) Then
        ValidBoehm = False
    ElseIf IsNull(BoehmRec.MODP) Then
        ValidBoehm = False
    ElseIf IsNull(BoehmRec.TOOL) Then
        ValidBoehm = False
    ElseIf IsNull(BoehmRec.SCED) Then
        ValidBoehm = False
    ElseIf IsNull(BoehmRec.RVOL) Then
        ValidBoehm = False
    ElseIf IsNull(BoehmRec.PI) Then
        ValidBoehm = False
    ElseIf IsNull(BoehmRec.Mode) Then
        ValidBoehm = False
    ElseIf IsNull(BoehmRec.TOTKDSI) Then
        ValidBoehm = False
    ElseIf IsNull(BoehmRec.ADJKDSI) Then
        ValidBoehm = False
    ElseIf IsNull(BoehmRec.NOMEFFORT) Then
        ValidBoehm = False
    ElseIf IsNull(BoehmRec.ESTEFFORT) Then
        ValidBoehm = False
    ElseIf IsNull(BoehmRec.ACTEFFORT) Then
        ValidBoehm = False
    Else
        ValidBoehm = True
    End If
    If ValidBoehm Then
```

```
    FindDBType = BoehmDB
    'MsgBox "Database is Boehm's original format."
    Exit Function
End If

'Check Sample database structure
AgiViewGet DBHandle, SampleRDS, SampleRec
If IsNull(SampleRec.Num) Then
    ValidSample = False
ElseIf IsNull(SampleRec.Type) Then
    ValidSample = False
ElseIf IsNull(SampleRec.RELY) Then
    ValidSample = False
ElseIf IsNull(SampleRec.DBSIZE) Then
    ValidSample = False
ElseIf IsNull(SampleRec.CPLX) Then
    ValidSample = False
ElseIf IsNull(SampleRec.EXTIME) Then
    ValidSample = False
ElseIf IsNull(SampleRec.STOR) Then
    ValidSample = False
ElseIf IsNull(SampleRec.VIRT) Then
    ValidSample = False
ElseIf IsNull(SampleRec.TURN) Then
    ValidSample = False
ElseIf IsNull(SampleRec.ACAP) Then
    ValidSample = False
ElseIf IsNull(SampleRec.AEXP) Then
    ValidSample = False
ElseIf IsNull(SampleRec.PCAP) Then
    ValidSample = False
ElseIf IsNull(SampleRec.VEXP) Then
    ValidSample = False
ElseIf IsNull(SampleRec.LEXP) Then
    ValidSample = False
ElseIf IsNull(SampleRec.MODP) Then
    ValidSample = False
ElseIf IsNull(SampleRec.TOOL) Then
    ValidSample = False
ElseIf IsNull(SampleRec.SCED) Then
    ValidSample = False
ElseIf IsNull(SampleRec.RVOL) Then
    ValidSample = False
ElseIf IsNull(SampleRec.PI) Then
    ValidSample = False
ElseIf IsNull(SampleRec.ACTKDSI) Then
    ValidSample = False
ElseIf IsNull(SampleRec.ACTEFFORT) Then
    ValidSample = False
Else
    ValidSample = True
End If
If ValidSample Then
    FindDBType = SampleDB
    'MsgBox "Database is in Sample format."
    Exit Function
End If

'Check Thesis database structure
AgiViewGet DBHandle, ThesisRDS, ThesisRec
AgiViewGet DBHandle, OldThesisRDS, OldThesisRec
ValidOldThesis = False
If IsNull(ThesisRec.BCNUM) Then
    ValidThesis = False
ElseIf IsNull(ThesisRec.Category) Then
    ValidThesis = False
ElseIf IsNull(ThesisRec.RELY) Then
    ValidThesis = False
ElseIf IsNull(ThesisRec.DBSIZE) Then
    ValidThesis = False
ElseIf IsNull(ThesisRec.CPLX) Then
    ValidThesis = False
ElseIf IsNull(ThesisRec.TIMEUTIL) Then
    ValidThesis = False
```

```
    ElseIf IsNull(ThesisRec.EXTIME) Then
        ValidThesis = False
    ElseIf IsNull(ThesisRec.MEMUTIL) Then
        ValidThesis = False
    ElseIf IsNull(ThesisRec.STOR) Then
        ValidThesis = False
    ElseIf IsNull(ThesisRec.VIRT) Then
        ValidThesis = False
    ElseIf IsNull(ThesisRec.TURN) Then
        ValidThesis = False
    ElseIf IsNull(ThesisRec.ACAP) Then
        ValidThesis = False
    ElseIf IsNull(ThesisRec.AEXP) Then
        ValidThesis = False
    ElseIf IsNull(ThesisRec.PCAP) Then
        ValidThesis = False
    ElseIf IsNull(ThesisRec.VEXP) Then
        ValidThesis = False
    ElseIf IsNull(ThesisRec.LEXP) Then
        ValidThesis = False
    ElseIf IsNull(ThesisRec.MODP) Then
        ValidThesis = False
    ElseIf IsNull(ThesisRec.TOOL) Then
        ValidThesis = False
    ElseIf IsNull(ThesisRec.SCED) Then
        ValidThesis = False
    ElseIf IsNull(ThesisRec.RVOL) Then
        ValidThesis = False
    ElseIf IsNull(ThesisRec.PI) Then
        ValidThesis = False
    ElseIf IsNull(ThesisRec.ENTROPY) Then
        ValidThesis = False
    ElseIf IsNull(ThesisRec.ACTKDSI) Then
        ValidThesis = False
    ElseIf IsNull(ThesisRec.ADDKDSI) Then
        ValidThesis = False
    ElseIf IsNull(ThesisRec.MODKDSI) Then
        ValidThesis = False
    ElseIf IsNull(ThesisRec.DELKDSI) Then
        ValidThesis = False
    ElseIf IsNull(ThesisRec.EDSI) Then
        ValidThesis = False
        If Not IsNull(OldThesisRec.ESDI) Then
            ValidOldThesis = True
        End If
    ElseIf IsNull(ThesisRec.ACTEFFORT) Then
        ValidThesis = False
    Else
        ValidThesis = True
    End If

    If ValidThesis Then
        FindDBType = ThesisDB
        'MsgBox "Database is in Thesis format."
        ThesisCategory = Trim(ThesisRec.Category)
        ThesisBCNum = Int(ThesisRec.BCNUM)
        more = AgiViewNext(DBHandle)
        Do While more
            AgiViewGet DBHandle, ThesisRDS, ThesisRec
            If ThesisCategory <> Trim(ThesisRec.Category) Or ThesisBCNum <> (Int(ThesisRec.BCNUM)
- 1) Then
                FindDBType = CompThesisDB
                'MsgBox "Database is in composite Thesis format."
                Exit Function
            End If
            ThesisBCNum = Int(ThesisRec.BCNUM)
            more = AgiViewNext(DBHandle)
        Loop
        Exit Function
    End If

    If ValidOldThesis Then
        FindDBType = ThesisDB
        MsgBox "Database is in Old Thesis format and needs conversion."
```

```
    End If

    MsgBox "Database is *.dbf or *.agi format but" + NL() + "record structure does not match
required formats."
    FindDBType = UnknownDB

End Function

Sub mnuCalBoehm_Click ()
    Select Case DBType
        Case BoehmDB
            FrmCalBm.Show
        Case SampleDB
            FrmSampCal.Show
        Case ThesisDB, CompThesisDB
            FrmThesCal.Show
        Case UnknownDB
    End Select
End Sub

Sub mnuCloseDB_Click ()
    AgiViewCloseAll
    mnuCloseDB.Enabled = False
    mnuViewGraph.Enabled = False
    mnuViewTable.Enabled = False
    mnuViewData.Visible = False
    mnuCalBoehm.Enabled = False
    mnuStatEval.Enabled = False
End Sub

Sub mnuCreateBoehmDB_Click ()
    frmDBCreate.Show
End Sub

Sub mnuCreateCompThesisDB_Click ()
    frmCompCreate.Show
End Sub

Sub mnuCreateThesisDB_Click ()
    frmThsCreate.Show
End Sub

Sub mnuDataConv_Click ()
    frmDataConv.Show
End Sub

Sub mnuExit_Click ()
    AgiViewCloseAll
    End
End Sub

Sub mnuOpenDB_Click ()
    AgiViewCloseAll     'Close any open DBs before open another
    On Error Resume Next
    CMDialog1.Flags = OFN_FILEMUSTEXIST
    CMDialog1.Action = DLG_FILE_OPEN
    If Err = 32755 Then           'CDERR_CANCEL
        Exit Sub
    End If
    On Error GoTo 0

    DBFileSpec = CMDialog1.Filename
    DBFileName = CMDialog1.Filetitle

    DBHandle = AgiFreeFile()
    AgiViewOpen DBHandle, DBFileSpec, "R"
    If AgiError() <> 0 Then
        Select Case AgiError()
            Case AGIE_DBOPEN
                'ErrMsg = "Database is already open."
            Case AGIE_NOTEXIST
                ErrMsg = "Database not found.  Please check path and filename."
            Case AGIE_BADFILE
                ErrMsg = "Specified file is corrupted or" + NL() + "is not a *.dbf or *.agi file
```

```
format."
            Case Else
                ErrMsg = AgiErrorText()
        End Select
        MsgBox ErrMsg + NL() + "Error#: " + Str(AgiError())

    Else
        more = AgiViewFirst(DBHandle)
        If more Then
            mnuCloseDB.Enabled = True
            DBType = FindDBType()
            Select Case DBType
                Case BoehmDB
                    mnuViewGraph.Enabled = True
                    mnuViewTable.Enabled = True
                    mnuViewData.Visible = True
                    mnuCalBoehm.Enabled = True
                    mnuStatEval.Enabled = True
                Case SampleDB
                    mnuViewGraph.Enabled = True
                    mnuViewTable.Enabled = True
                    mnuViewData.Visible = False
                    mnuCalBoehm.Enabled = True
                    mnuStatEval.Enabled = False
                Case ThesisDB, CompThesisDB
                    mnuViewGraph.Enabled = True
                    mnuViewTable.Enabled = True
                    mnuViewData.Visible = False
                    mnuCalBoehm.Enabled = True
                    mnuStatEval.Enabled = True
                Case UnknownDB
                    ErrMsg = "Format of selected database is not recognizeable." + NL()
                    ErrMsg = ErrMsg + "Only viewing is permitted."
                    MsgBox ErrMsg
                    mnuViewGraph.Enabled = False
                    mnuViewTable.Enabled = True
                    mnuViewData.Visible = False
                    mnuCalBoehm.Enabled = False
                    mnuStatEval.Enabled = False
            End Select
        Else
            AgiViewClose DBHandle
            ErrMsg = "Database exists but has zero records."
            MsgBox ErrMsg
        End If
    End If
End Sub

Sub mnuStatEval_Click ()
    Select Case DBType
        Case BoehmDB
            frmStatBoehm.Show
        Case SampleDB
        Case ThesisDB
            frmThesStat.Show
        Case CompThesisDB
            frmThesStat.Show
        Case UnknownDB
    End Select
End Sub

Sub mnuViewData_Click ()
    FrmBoehm.Show
End Sub

Sub mnuViewGraph_Click ()
    Select Case DBType
        Case BoehmDB
            frmBoehmGraph.Show
        Case SampleDB
            frmBoehmGraph.Show
        Case ThesisDB
            frmThesGraph.Show
        Case CompThesisDB
```

```
            frmCompGraph.Show
        Case UnknownDB
    End Select
End Sub

Sub mnuViewTable_Click ()
    Select Case DBType
        Case BoehmDB
            frmBoehmTable.txtDBName.Text = DBFileSpec
            'frmBoehmTable.Table1.FileName = DBFileSpec
            frmBoehmTable.Table1.ViewHandle = DBHandle
            frmBoehmTable.Show
        Case SampleDB
            frmSampTable.txtDBName.Text = DBFileSpec
            frmSampTable.Table1.FileName = DBFileSpec
            frmSampTable.Table1.ViewHandle = DBHandle
            frmSampTable.Show
        Case ThesisDB, CompThesisDB
            frmThesTable.txtDBName.Text = DBFileSpec
            frmThesTable.Table1.ViewHandle = DBHandle
            frmThesTable.Show
        Case UnknownDB
            frmTableDB.txtDBName.Text = DBFileSpec
            frmTableDB.Table1.ViewHandle = DBHandle
            frmTableDB.Show
    End Select
End Sub
```

Code for SampCal Form

```
Option Explicit

Sub cmdCalibrate_Click ()
    Dim Mode        As String
    Dim ModeCount   As Integer
    Dim RecCount    As Integer

    Dim CoeffOnly   As Single
    Dim ExpOnly     As Single
    Dim Coefficient As Single
    Dim Exponent    As Single

    Dim TempQ       As Double
    Dim SumMMQ      As Double
    Dim SumQ2       As Double
    Dim TempLog     As Double
    Dim TempDiv     As Double
    Dim a0          As Integer
    Dim a1          As Double
    Dim a2          As Double
    Dim d0          As Double
    Dim d1          As Double

    If optOrganic.Value = True Then
        Mode = "ORG"
        ExpOnly = 1.05
    ElseIf optSemiD.Value = True Then
        Mode = "SD"
        ExpOnly = 1.12
    ElseIf optEmbedded.Value = True Then
        Mode = "E"
        ExpOnly = 1.2
    ElseIf optAllData.Value = True Then
        Mode = "ALL"
        ExpOnly = 1.16
    Else
        Mode = "UNK"
        ExpOnly = 1.16
    End If

    SampleHandle = DBHandle
    more = AgiViewFirst(SampleHandle)

    SumMMQ = 0#
    SumQ2 = 0#
    ModeCount = 0
    txtModeTotal.Text = Format$(ModeCount, "###0")
    RecCount = 0
    a1 = 0#
    a2 = 0#
    d0 = 0#
    d1 = 0#

    Do While more
        RecCount = RecCount + 1
        txtRecTotal.Text = Format$(RecCount, "###0")
        AgiViewGet SampleHandle, SampleRDS, SampleRec
        If Trim(SampleRec.Type) = Trim(Mode) Or Mode = "ALL" Then
            ModeCount = ModeCount + 1
            txtModeTotal.Text = Format$(ModeCount, "###0")

            'Calculate sums for Coefficient only
            TempQ = SampleRec.PI * (SampleRec.ACTKDSI ^ ExpOnly)
            SumMMQ = SumMMQ + SampleRec.ACTEFFORT * TempQ
            SumQ2 = SumQ2 + TempQ * TempQ

            'Calculate sums for Coefficient and Exponent
            TempLog = Log(SampleRec.ACTKDSI)
            TempDiv = Log(SampleRec.ACTEFFORT / SampleRec.PI)
            a1 = a1 + TempLog
            a2 = a2 + TempLog * TempLog
            d0 = d0 + TempDiv
```

```
            d1 = d1 + TempDiv * TempLog

        End If
        more = AgiViewNext(SampleHandle)
    Loop

    If ModeCount > 0 Then
        CoeffOnly = SumMMQ / SumQ2
        a0 = ModeCount
        Coefficient = Exp((a2 * d0 - a1 * d1) / (a0 * a2 - a1 * a1))
        Exponent = (a0 * d1 - a1 * d0) / (a0 * a2 - a1 * a1)
    End If

    txtCoeffOnly.Text = Format$(CoeffOnly, "0.0000")
    txtCoefficient.Text = Format$(Coefficient, "0.0000")
    txtExponent.Text = Format$(Exponent, "0.0000")
End Sub

Sub cmdCancel_Click ()
    Unload FrmSampCal
End Sub

Sub Form_Load ()
    txtDBName.Text = DBFileSpec
End Sub
```

Code for SampTbl Form

```
Option Explicit

Sub cmdCancel_Click ()
    Unload frmSampTable
End Sub
```

Code for ThesCal Form

```
Option Explicit

Sub cmdCalibrate_Click ()
    Dim Category      As String
    Dim CategoryCount As Integer
    Dim RecCount      As Integer

    Dim CoeffOnly   As Single
    Dim ExpOnly     As Single
    Dim Coefficient As Single
    Dim Exponent    As Single
    Dim BCT         As Double

    Dim TempQ       As Double
    Dim SumMMQ      As Double
    Dim SumQ2       As Double
    Dim TempLog     As Double
    Dim TempDiv     As Double
    Dim a0          As Integer
    Dim a1          As Double
    Dim a2          As Double
    Dim d0          As Double
    Dim d1          As Double

    If optCID.Value = True Then
        Category = "CID"
        ExpOnly = 1.05
    ElseIf optNAV.Value = True Then
        Category = "NAV"
        ExpOnly = 1.12
    ElseIf optCAV.Value = True Then
        Category = "CAV"
        ExpOnly = 1.2
    ElseIf optECS.Value = True Then
        Category = "ECS"
        ExpOnly = 1.28
    ElseIf optOFF.Value = True Then
        Category = "OFF"
        ExpOnly = 1.36
    ElseIf optAllData.Value = True Then
        Category = "ALL"
        ExpOnly = 1.16
    Else
        Category = "UNK"
        ExpOnly = 1.16
    End If

    ThesisHandle = DBHandle
    more = AgiViewFirst(ThesisHandle)

    SumMMQ = 0#
    SumQ2 = 0#
    CategoryCount = 0
    txtCategoryTotal.Text = Format$(CategoryCount, "###0")
    RecCount = 0
    a1 = 0#
    a2 = 0#
    d0 = 0#
    d1 = 0#

    Do While more
        RecCount = RecCount + 1
        txtRecTotal.Text = Format$(RecCount, "###0")
        AgiViewGet ThesisHandle, ThesisRDS, ThesisRec
        If Trim(ThesisRec.Category) = Trim(Category) Or Category = "ALL" Then
            CategoryCount = CategoryCount + 1
            txtCategoryTotal.Text = Format$(CategoryCount, "###0")

            'Calculate sums for Coefficient only
            BCT = ThesisRec.EDSI / ThesisRec.ACTKDSI
            TempQ = ThesisRec.PI * (ThesisRec.ACTKDSI ^ ExpOnly) * BCT
            SumMMQ = SumMMQ + ThesisRec.ACTEFFORT * TempQ
```

```
            SumQ2 = SumQ2 + TempQ * TempQ

            'Calculate sums for Coefficient and Exponent
            TempLog = Log(ThesisRec.ACTKDSI)
            TempDiv = Log(ThesisRec.ACTEFFORT / (ThesisRec.PI * BCT))
            a1 = a1 + TempLog
            a2 = a2 + TempLog * TempLog
            d0 = d0 + TempDiv
            d1 = d1 + TempDiv * TempLog

        End If
        more = AgiViewNext(ThesisHandle)
    Loop

    If CategoryCount > 0 Then
        CoeffOnly = SumMMQ / SumQ2
        a0 = CategoryCount
        Coefficient = Exp((a2 * d0 - a1 * d1) / (a0 * a2 - a1 * a1))
        Exponent = (a0 * d1 - a1 * d0) / (a0 * a2 - a1 * a1)
    End If

    txtCoeffOnly.Text = Format$(CoeffOnly, "0.0000")
    txtCoefficient.Text = Format$(Coefficient, "0.0000")
    txtExponent.Text = Format$(Exponent, "0.0000")
End Sub

Sub cmdCancel_Click ()
    Unload frmThesCal
End Sub

Sub Form_Load ()
    txtDBName.Text = DBFileSpec
End Sub
```

Code for ThesStat Form

```
Option Explicit

Sub cmdCalc_Click ()
    Dim Category      As String
    Dim RecCategory   As String
    Dim CategoryCount As Integer
    Dim RecCount      As Integer

    Dim PI              As Double
    Dim BaseCoefficient As Single
    Dim BaseExponent    As Single
    Dim PredCoefficient As Single
    Dim PredExponent    As Single
    Dim EstEffort       As Double
    Dim LogEstEffort    As Double
    Dim ACTEFFORT       As Single
    Dim LogActEffort    As Double
    Dim ActEffMean      As Double
    Dim LogActEffMean   As Double

    Dim ThesisHandle As Single

    Dim SumActEff   As Double
    Dim SumLogActEff As Double
    Dim SSE         As Double
    Dim LogSSE      As Double
    Dim LogSSTO     As Double
    Dim SumMRE      As Double
    Dim SumLogMRE   As Double
    Dim R2          As Single
    Dim RRMS        As Single
    Dim MRE         As Single
    Dim LogMRE      As Single
    Dim MREMean     As Single
    Dim PredLevel   As Single
    Const MRELimit = .25
    Dim ImproveLevel As Single

    If optCID.Value = True Then
        Category = "CID"
    ElseIf optNAV.Value = True Then
        Category = "NAV"
    ElseIf optCAV.Value = True Then
        Category = "CAV"
    ElseIf optECS.Value = True Then
        Category = "ECS"
    ElseIf optOFF.Value = True Then
        Category = "OFF"
    ElseIf optAllData.Value = True Then
        Category = "ALL"
    Else
        Category = "UNK"
    End If
    Category = Trim(Category)

    BaseCoefficient = Val(txtBaseCoeff.Text)
    BaseExponent = Val(txtBaseExp.Text)
    PredCoefficient = Val(txtPredCoeff.Text)
    PredExponent = Val(txtPredExp.Text)

    ThesisHandle = DBHandle
    'AgiViewOpen BoehmHandle, "boehm's.dbf", "R"
    'Debug.Print AgiErrorText()

    'Find Average of Actual Efforts in Real and Log domains
    SumActEff = 0#
    SumLogActEff = 0#
    CategoryCount = 0

    more = AgiViewFirst(ThesisHandle)
    Do While more
        AgiViewGet ThesisHandle, "CATEGORY(S)", RecCategory
```

```
        If Category = Trim(RecCategory) Or Category = "ALL" Then
            CategoryCount = CategoryCount + 1
            'txtNumMode.Text = Format$(CategoryCount, "###0")
            AgiViewGet ThesisHandle, "ACTEFFORT(F)", ACTEFFORT
            SumActEff = SumActEff + ACTEFFORT
            SumLogActEff = SumLogActEff + Log(ACTEFFORT)
        End If
        more = AgiViewNext(ThesisHandle)
    Loop
    If CategoryCount > 0 Then
        ActEffMean = SumActEff / CategoryCount
        LogActEffMean = SumLogActEff / CategoryCount
    Else
        MsgBox "No Records found with matching Mode."
        Exit Sub
    End If

    'Caluclate Statistics
    SSE = 0#
    LogSSE = 0#
    LogSSTO = 0#
    SumMRE = 0#
    SumLogMRE = 0#
    PredLevel = 0!
    ImproveLevel = 0!
    RecCount = 0
    CategoryCount = 0

    more = AgiViewFirst(ThesisHandle)
    txtNumMode.Text = Format$(CategoryCount, "###0")
    Do While more
        RecCount = RecCount + 1
        txtNumRec.Text = Format$(RecCount, "###0")
        AgiViewGet ThesisHandle, ThesisRDS, ThesisRec
        If Category = Trim(ThesisRec.Category) Or Category = "ALL" Then
            CategoryCount = CategoryCount + 1
            txtNumMode.Text = Format$(CategoryCount, "###0")
            'PI = BoehmRec.RELY * BoehmRec.DBSIZE * BoehmRec.CPLX * BoehmRec.EXTIME
            'PI = PI * BoehmRec.STOR * BoehmRec.VIRT * BoehmRec.TURN * BoehmRec.ACAP
            'PI = PI * BoehmRec.AEXP * BoehmRec.PCAP * BoehmRec.VEXP * BoehmRec.LEXP
            'PI = PI * BoehmRec.MODP * BoehmRec.TOOL * BoehmRec.SCED * BoehmRec.RVOL
            EstEffort = ThesisRec.PI * PredCoefficient * (ThesisRec.ACTKDSI ^ PredExponent) *
(ThesisRec.EDSI / ThesisRec.ACTKDSI)
            LogEstEffort = Log(EstEffort)
            MRE = Abs(ThesisRec.ACTEFFORT - EstEffort)
            LogMRE = Abs(Log(ThesisRec.ACTEFFORT) - LogEstEffort)

            'Calculate sums
            SSE = SSE + MRE ^ 2
            LogSSE = LogSSE + LogMRE ^ 2
            LogSSTO = LogSSTO + (Log(ThesisRec.ACTEFFORT) - LogActEffMean) ^ 2
            SumMRE = SumMRE + MRE
            If MRE / ThesisRec.ACTEFFORT <= MRELimit Then
                PredLevel = PredLevel + 1#
            End If
            If MRE < Abs(ThesisRec.ACTEFFORT - (ThesisRec.PI * BaseCoefficient *
(ThesisRec.ACTKDSI ^ BaseExponent) * (ThesisRec.EDSI / ThesisRec.ACTKDSI))) Then
                ImproveLevel = ImproveLevel + 1#
            End If
        End If
        more = AgiViewNext(ThesisHandle)
    Loop
    'AgiViewClose BoehmHandle

    If CategoryCount > 0 Then
        R2 = 1 - (LogSSE * (CategoryCount - 1)) / (LogSSTO * (CategoryCount - 2))
        RRMS = (Sqr(SSE / CategoryCount)) / ActEffMean
        MREMean = SumMRE / CategoryCount
        PredLevel = (PredLevel / CategoryCount) * 100
        ImproveLevel = (ImproveLevel / CategoryCount) * 100
    End If

    Text3.Text = Format$(R2, "0.0000")
    Text4.Text = Format$(RRMS, "0.000")
```

```
    Text5.Text = Format$(MREMean, "0.00")
    Text6.Text = Format$(PredLevel, "#0.0") + "%"
    Text9.Text = Format$(ImproveLevel, "#0.0") + "%"
End Sub

Sub cmdCancel_Click ()
    Unload frmThesStat
End Sub
```

Code for ThesTabl Form

```
Option Explicit

Sub cmdCancel_Click ()
    Unload frmThesTable
End Sub
```

Code for ThesYAxs Form

```
Option Explicit

Sub cmdCancel_Click ()
    Unload frmYAxis
End Sub

Sub cmdOK_Click ()
    NormState = Norm_Off
    frmThesGraph!cmdNorm.Enabled = False
    If optEXTIME.Value = True Then
        YPick = Executable_Time
        frmThesGraph!Graph1.LeftTitle = "EXTIME"
    ElseIf optTimeUtil.Value = True Then
        YPick = Time_Util
        frmThesGraph!Graph1.LeftTitle = "Time %"
    ElseIf optStor.Value = True Then
        YPick = Storage
        frmThesGraph!Graph1.LeftTitle = "STOR"
    ElseIf optMemUtil.Value = True Then
        YPick = Mem_Util
        frmThesGraph!Graph1.LeftTitle = "Mem %"
    ElseIf optPi.Value = True Then
        YPick = Pi_Mult
        frmThesGraph!Graph1.LeftTitle = "PI"
    ElseIf optAexp.Value = True Then
        YPick = Analyst_Experience
        frmThesGraph!Graph1.LeftTitle = "AEXP"
    ElseIf optLexp.Value = True Then
        YPick = Lang_Experience
        frmThesGraph!Graph1.LeftTitle = "LEXP"
    ElseIf optActKDSI.Value = True Then
        YPick = Actual_KDSI
        frmThesGraph!Graph1.LeftTitle = "Actual KDSI"
    ElseIf optEDSI.Value = True Then
        YPick = Equivalent_DSI
        frmThesGraph!Graph1.LeftTitle = "Equiv KDSI"
    ElseIf optActEffort.Value = True Then
        frmThesGraph!cmdNorm.Enabled = True
        YPick = Actual_Effort
        frmThesGraph!Graph1.LeftTitle = "MM"
    End If

    Unload frmYAxis
End Sub

Sub Form_Load ()
    Select Case YPick
        Case Executable_Time
            optEXTIME.Value = True
        Case Time_Util
            optTimeUtil.Value = True
        Case Storage
            optStor.Value = True
        Case Mem_Util
            optMemUtil.Value = True
        Case Pi_Mult
            optPi.Value = True
        Case Analyst_Experience
            optAexp.Value = True
        Case Lang_Experience
            optLexp.Value = True
        Case Actual_KDSI
            optActKDSI.Value = True
        Case Equivalent_DSI
            optEDSI.Value = True
        Case Actual_Effort
            optActEffort.Value = True
    End Select
End Sub
```

Code for ThsCreat Form

```
Option Explicit
Dim ThesisFileSpec As String

Sub cmdCancel_Click ()
    'cmdClear_Click
    Unload frmThsCreate 'Close SampleHandle is in Unload
End Sub

Sub cmdClear_Click ()
    While AgiViewFirst(ThesisHandle)
        AgiViewDelete ThesisHandle
        If AgiError() <> 0 Then
            MsgBox "Record deletion error: " + AgiErrorText()
        End If
    Wend
    cmdCreate.Enabled = True
    cmdClear.Enabled = False
    cmdSave.Enabled = False
End Sub

Sub cmdCreate_Click ()
    Dim NumBC           As Integer
    Dim NumRec          As Integer
    Dim EffSDPercent    As Single
    Dim SD              As Single
    Dim EAF             As Double
    Dim EstEffort       As Single
    Dim PrevBC          As ThesisType
    Dim LearnEffect     As Double
    'Dim EstEffort       As Single

    Randomize           'Reseeds random number generator
    SD = Val(txtFactorSD.Text)

    cmdClear_Click      'Clear existing records if any

    'Create record from Development Data
    ThesisRec.BCNUM = 1
    If optCID.Value = True Then
        ThesisRec.Category = "CID"
    ElseIf optNAV.Value = True Then
        ThesisRec.Category = "NAV"
    ElseIf optCAV.Value = True Then
        ThesisRec.Category = "CAV"
    ElseIf optECS.Value = True Then
        ThesisRec.Category = "ECS"
    ElseIf optOFF.Value = True Then
        ThesisRec.Category = "OFF"
    End If
    ThesisRec.RELY = NDEV(1, SD)
    ThesisRec.DBSIZE = NDEV(1, SD)
    ThesisRec.CPLX = NDEV(1, SD)
    ThesisRec.TIMEUTIL = Val(txtDevTime.Text)
    ThesisRec.EXTIME = FindEXTIME(Val(ThesisRec.TIMEUTIL))
    ThesisRec.MEMUTIL = Val(txtDevMem.Text)
    ThesisRec.STOR = FindSTOR(Val(ThesisRec.MEMUTIL))
    ThesisRec.VIRT = NDEV(1, SD)
    ThesisRec.TURN = NDEV(1, SD)
    ThesisRec.ACAP = NDEV(1, SD)
    ThesisRec.AEXP = NDEV(1, SD)
    ThesisRec.PCAP = NDEV(1, SD)
    ThesisRec.VEXP = NDEV(1, SD)
    ThesisRec.LEXP = NDEV(1, SD)
    ThesisRec.MODP = NDEV(1, SD)
    ThesisRec.TOOL = NDEV(1, SD)
    ThesisRec.SCED = NDEV(1, SD)
    ThesisRec.RVOL = NDEV(1, SD)
    EAF = ThesisRec.RELY * ThesisRec.DBSIZE * ThesisRec.CPLX * ThesisRec.EXTIME
    EAF = EAF * ThesisRec.STOR * ThesisRec.VIRT * ThesisRec.TURN * ThesisRec.ACAP
    EAF = EAF * ThesisRec.AEXP * ThesisRec.PCAP * ThesisRec.VEXP * ThesisRec.LEXP
    EAF = EAF * ThesisRec.MODP * ThesisRec.TOOL * ThesisRec.SCED * ThesisRec.RVOL
    ThesisRec.PI = EAF
```

```
    ThesisRec.ENTROPY = 1#
    ThesisRec.ACTKDSI = Val(txtDevKDSI.Text)
    ThesisRec.ADDKDSI = ThesisRec.ACTKDSI * (Val(txtPerAdd.Text) / 100) * NDEV(1,
Val(txtPerSD.Text) / 100)
    ThesisRec.MODKDSI = ThesisRec.ACTKDSI * (Val(txtPerMod.Text) / 100) * NDEV(1,
Val(txtPerSD.Text) / 100)
    ThesisRec.DELKDSI = ThesisRec.ACTKDSI * (Val(txtPerDel.Text) / 100) * NDEV(1,
Val(txtPerSD.Text) / 100)
    ThesisRec.EDSI = FindEDSI(Val(ThesisRec.ADDKDSI), Val(ThesisRec.MODKDSI),
Val(ThesisRec.DELKDSI))
    EstEffort = EAF * Val(txtCoefficient.Text) * (ThesisRec.ACTKDSI ^ Val(txtExponent.Text)) *
(ThesisRec.EDSI / ThesisRec.ACTKDSI)
    'EffSDPercent is the % of SD at the EDSI value
    EffSDPercent = Val(txtEffortSD.Text) / ThesisRec.EDSI
    ThesisRec.ACTEFFORT = Exp(Log(EstEffort) + NDEV(0, Log(1 + EffSDPercent)))
    AgiViewAdd ThesisHandle, ThesisRDS, ThesisRec
    If AgiError() <> 0 Then
        MsgBox "Could not create first record."
        cmdClear_Click
        Exit Sub
    End If

    'Create Time series records for remaining block changes
    PrevBC = ThesisRec      'Hold prior years data
    Screen.MousePointer = HourGlass
    For NumRec = 2 To Int(Val(txtBCNum.Text))
        ThesisRec = PrevBC      'Intialize this block change data
        ThesisRec.BCNUM = NumRec
        If PrevBC.TIMEUTIL < 95# Then
            'Increase at KDSI growth rate
            ThesisRec.TIMEUTIL = PrevBC.TIMEUTIL * (1# + ((PrevBC.ADDKDSI - PrevBC.DELKDSI) /
PrevBC.ACTKDSI))
        Else
            'Increase half remaining
            ThesisRec.TIMEUTIL = PrevBC.TIMEUTIL + (.5 * (100# - PrevBC.TIMEUTIL))
        End If
        If ThesisRec.TIMEUTIL > 100# Then
            MsgBox "The Throughput growth has exceed 100% capacity" + NL() + "in block change " +
Str(NumRec) + "."
            Exit For
        End If
        ThesisRec.EXTIME = FindEXTIME(Val(ThesisRec.TIMEUTIL))
        If PrevBC.MEMUTIL < 95# Then
            'Increase at KDSI growth rate
            ThesisRec.MEMUTIL = PrevBC.MEMUTIL * (1# + ((PrevBC.ADDKDSI - PrevBC.DELKDSI) /
PrevBC.ACTKDSI))
        Else
            'Increase half remaining
            ThesisRec.MEMUTIL = PrevBC.MEMUTIL + (.5 * (100# - PrevBC.MEMUTIL))    'add half
remaining
        End If
        If ThesisRec.MEMUTIL > 100# Then
            MsgBox "The memory growth has exceed 100% capacity" + NL() + "in block change " +
Str(NumRec) + "."
            Exit For
        End If
        ThesisRec.STOR = FindSTOR(Val(ThesisRec.MEMUTIL))
        LearnEffect = FindLearn(Int(ThesisRec.BCNUM))
        'Spread Learning Effect to AEXP and LEXP
        ThesisRec.AEXP = PrevBC.AEXP * Sqr(LearnEffect)
        ThesisRec.LEXP = PrevBC.LEXP * Sqr(LearnEffect)
        EAF = ThesisRec.RELY * ThesisRec.DBSIZE * ThesisRec.CPLX * ThesisRec.EXTIME
        EAF = EAF * ThesisRec.STOR * ThesisRec.VIRT * ThesisRec.TURN * ThesisRec.ACAP
        EAF = EAF * ThesisRec.AEXP * ThesisRec.PCAP * ThesisRec.VEXP * ThesisRec.LEXP
        EAF = EAF * ThesisRec.MODP * ThesisRec.TOOL * ThesisRec.SCED * ThesisRec.RVOL
        ThesisRec.PI = EAF

        ThesisRec.ACTKDSI = PrevBC.ACTKDSI + PrevBC.ADDKDSI - PrevBC.DELKDSI
        ThesisRec.ADDKDSI = ThesisRec.ACTKDSI * (Val(txtPerAdd.Text) / 100) * NDEV(1,
Val(txtPerSD.Text) / 100)
        ThesisRec.MODKDSI = ThesisRec.ACTKDSI * (Val(txtPerMod.Text) / 100) * NDEV(1,
Val(txtPerSD.Text) / 100)
        ThesisRec.DELKDSI = ThesisRec.ACTKDSI * (Val(txtPerDel.Text) / 100) * NDEV(1,
Val(txtPerSD.Text) / 100)
```

```
        ThesisRec.EDSI = FindEDSI(Val(ThesisRec.ADDKDSI), Val(ThesisRec.MODKDSI),
Val(ThesisRec.DELKDSI))
        EstEffort = EAF * Val(txtCoefficient.Text) * (ThesisRec.ACTKDSI ^ Val(txtExponent.Text)) *
(ThesisRec.EDSI / ThesisRec.ACTKDSI)
        'EffSDPercent is the % of SD at the EDSI value
        EffSDPercent = Val(txtEffortSD.Text) / ThesisRec.EDSI
        ThesisRec.ACTEFFORT = Exp(Log(EstEffort) + NDEV(0, Log(1 + EffSDPercent)))
        AgiViewAdd ThesisHandle, ThesisRDS, ThesisRec
        If AgiError() <> 0 Then
            MsgBox "File Error: Could not add block change " + Str(NumRec) + "."
            cmdClear_Click
            Exit Sub
        End If
        PrevBC = ThesisRec      'Hold previous block change data
    Next NumRec
    cmdCreate.Enabled = False
    cmdClear.Enabled = True
    cmdSave.Enabled = True
    Screen.MousePointer = Default
End Sub

Sub cmdSave_Click ()
    Dim SaveHandle As Integer
    Dim Filetitle As String

    'Get new FileSpec
    On Error Resume Next
    CMDialog1.DefaultExt = "AGI"
    CMDialog1.Flags = OFN_PATHMUSTEXIST + OFN_OVERWRITEPROMPT + OFN_HIDEREADONLY +
OFN_EXTENSIONDIFFERENT
    CMDialog1.Action = DLG_FILE_SAVE
    If Err = 32755 Then         'user selcted cancel button
        Exit Sub
    End If
    On Error GoTo 0

    Debug.Print "Save Filetitle is " + CMDialog1.Filetitle + "."
    'Test to see if new file is already open or exists
    Screen.MousePointer = HourGlass
    If CMDialog1.Filetitle = frmMain!CMDialog1.Filetitle Then
        AgiViewClose DBHandle               'Close Open Copy
        Kill CMDialog1.Filename             'Delete Previous open copy
        frmMain!mnuCloseDB.Enabled = False  'Reset menu selection
        frmMain!mnuViewGraph.Enabled = False
        frmMain!mnuViewTable.Enabled = False
        frmMain!mnuViewData.Visible = False
        frmMain!mnuCalBoehm.Enabled = False
        frmMain!mnuStatEval.Enabled = False
    ElseIf CMDialog1.Filetitle = "THSFILE.AGI" Then
        Exit Sub          'File update will be automatic upon Cancel_Click
    ElseIf Dir$(CMDialog1.Filename) <> "" Then     'If File exists
        Kill CMDialog1.Filename             'Delete unopened coy
    End If

    ' Open blank file to copy into
    SaveHandle = AgiFreeFile()
    Filetitle = CMDialog1.Filetitle
    AgiViewOpen SaveHandle, Filetitle, "CAUR"
    If AgiError() <> 0 Then
        MsgBox "File Creation Error: " + AgiErrorText()
    End If

    ' Copy current file into blank copy for saving
    more = AgiViewFirst(ThesisHandle)
    While more
        AgiViewGet ThesisHandle, ThesisRDS, ThesisRec
        AgiViewAdd SaveHandle, ThesisRDS, ThesisRec
        If AgiError() <> 0 Then
            MsgBox "Error adding record to save file." + NL() + "File results undetermined."
            AgiViewClose (SaveHandle)
            Screen.MousePointer = Default
            Exit Sub
        End If
        more = AgiViewNext(ThesisHandle)
```

```
    Wend
    AgiViewClose (SaveHandle)       'SampleHandle is remains open
    Screen.MousePointer = Default
End Sub

Function FindEDSI (ADDKDSI As Double, MODKDSI As Double, DELKDSI As Double) As Double
    'Solution is found by weighting the KDSI values
    'These weightings are from the Softcost-R Manual page R-83
    'Note the wieghtings for lines and modules have been added so as to include
    'the effect of altering both the line and the module
    FindEDSI = .53 * ADDKDSI + (.27 + .24) * MODKDSI + (.15 + .11) * DELKDSI
End Function

Function FindEXTIME (PerTime As Double) As Double
    If PerTime <= 65# Then
        FindEXTIME = 1#
    Else
        FindEXTIME = 1.82 * (PerTime / 100#) ^ 1.305
    End If
End Function

Function FindLearn (BCNUM As Integer) As Double
    'This function assumes the realtionship holds true for 6 block changes.
    'But SYSCON states relationship holds for 6 years.
    'Future may need to adjust for years if block changes are not 1 year
    'The exponent comes from SYSCON, Thesis TAble 5
    If BCNUM >= 2 And BCNUM <= 6 Then
        'No coefficient => normalize to 1.0
        'Note Effect is ratio to previous BC and not the first BC
        FindLearn = (BCNUM ^ (-.375)) / ((BCNUM - 1) ^ (-.375))
    Else
        FindLearn = 1#
    End If
End Function

Function FindSTOR (PerMem As Double) As Double
    If PerMem <= 65# Then
        FindSTOR = 1#
    Else
        FindSTOR = 1.94 * (PerMem / 100#) ^ 1.425
    End If
End Function

Sub Form_Load ()
    Dim Filetitle As String
    Dim FileSpec As String

    'Open new file (agi formats only).
    ThesisHandle = AgiFreeFile()
    Filetitle = "THSFILE.AGI"
    AgiViewOpen ThesisHandle, Filetitle, "CAUR"
    If AgiError() <> 0 Then
        MsgBox "File Open Error: " + AgiErrorText()
    End If
    Table1.ViewHandle = ThesisHandle

    'Enable Appropriate Buttons
    more = AgiViewFirst(ThesisHandle)
    If more Then
        cmdCreate.Enabled = False
    Else
        cmdClear.Enabled = False
        cmdSave.Enabled = False
    End If

    'Set FileSpec for Save Dialog Box
    If Right(File1.Path, 1) <> "\" Then
        FileSpec = File1.Path + "\" + Filetitle
    Else
        FileSpec = File1.Path + Filetitle
    End If
    CMDialog1.Filename = FileSpec
End Sub
```

```
Sub Form_Unload (Cancel As Integer)
    AgiViewClose ThesisHandle
End Sub

Sub txtBCNum_LostFocus ()
    Dim BCNUM As Integer
    BCNUM = Int(Val(txtBCNum.Text))
    If BCNUM > 0 Then
        txtBCNum.Text = Str(BCNUM)
    Else
        ErrMsg = "Please select a positve value for" + NL()
        ErrMsg = ErrMsg + "the number of records."
        MsgBox ErrMsg
        txtBCNum.Text = "10"
        txtBCNum.SetFocus
    End If
End Sub
```

Code for ThsGraph Form

```
Option Explicit

Sub cmdCancel_Click ()
    Graph1.DataReset = G_All_Data
    Unload frmThesGraph
End Sub

Sub cmdNorm_Click ()
    If NormState = Norm_Off Then
        cmdNorm.Caption = "Normalize Off"
        NormState = Norm_On
    Else
        cmdNorm.Caption = "Normalize On"
        NormState = Norm_Off
    End If

    YPick = Actual_Effort
    LoadThesisData NormState, YPick

    Graph1.DrawMode = G_Draw
    If FileGraph = 1 Then
        Graph1.DrawMode = G_Write
    End If
End Sub

Sub cmdYAxis_Click ()
    frmYAxis.Show Modal

    LoadThesisData NormState, YPick

    Graph1.DrawMode = G_Draw
    If FileGraph = 1 Then
        Graph1.DrawMode = G_Write
    End If
End Sub

Sub Form_Load ()
    txtDBName.Text = DBFileSpec

    'set filename for saving graph
    Graph1.ImageFile = "thesgrph"

    Screen.MousePointer = HourGlass
    WindowState = Maximized
    NormState = Norm_Off
    Graph1.LeftTitle = "MM"
    Graph1.BottomTitle = "Block Change Number"
    Graph1.DrawMode = G_Draw

    YPick = Actual_Effort
    LoadThesisData NormState, YPick
    Screen.MousePointer = HourGlass

    Graph1.DrawMode = G_Draw
    If FileGraph = 1 Then
        Graph1.DrawMode = G_Write
    End If
    If FileGraph = 1 Then
        Graph1.DrawMode = G_Write
    End If
    Screen.MousePointer = Default
End Sub

Sub LoadThesisData (Normalize As Integer, YChoice As Integer)
    Dim BCCount As Integer

    Graph1.NumPoints = AgiViewCount(DBHandle)
    Graph1.IndexStyle = 1   'Enhanced index style permits access to
                            'arrays for holding data.

    'Set mode characteristics
    Graph1.ThisSet = 1
```

```
    Graph1.ThisPoint = 1

    'Array subscripting
    '       1 = Comm / ID
    '       2 = Navigation Sensors
    '       3 = Core Avionics
    '       4 = Electronic Combat Systems
    '       5 = Offensive Sensors
    more = AgiViewFirst(DBHandle)
    If more Then
        AgiViewGet DBHandle, ThesisRDS, ThesisRec
        Select Case Trim(ThesisRec.Category)
            Case "CID"
                Graph1.LegendText = "C/ID"
                Graph1.SymbolData = G_CROSS_PLUS
                Graph1.ColorData = G_BLUE
            Case "NAV"
                Graph1.LegendText = "NAV"
                Graph1.SymbolData = G_CROSS_TIMES
                Graph1.ColorData = G_GREEN
            Case "CAV"
                Graph1.LegendText = "CORE"
                Graph1.SymbolData = G_TRIANGLE_UP
                Graph1.ColorData = G_RED
            Case "ECS"
                Graph1.LegendText = "E C"
                Graph1.SymbolData = G_DIAMOND
                Graph1.ColorData = G_BLACK
            Case "OFF"
                Graph1.LegendText = "OFF"
                Graph1.SymbolData = G_SQUARE
                Graph1.ColorData = G_BROWN
        End Select
    End If

    'Load the data to the graph
    BCCount = 0
    more = AgiViewFirst(DBHandle)
    Screen.MousePointer = HourGlass
    Do While more
        AgiViewGet DBHandle, ThesisRDS, ThesisRec
        BCCount = BCCount + 1
        Graph1.ThisPoint = BCCount
        Graph1.XPosData = BCCount
        If Normalize Then
            'Graph1.GraphData = ThesisRec.ACTEFFORT / ThesisRec.PI
            'Note below use default coefficient and exponent
            Graph1.GraphData = ThesisRec.PI * 2.85 * (ThesisRec.ACTKDSI ^ 1.15) * (ThesisRec.EDSI
/ ThesisRec.ACTKDSI)
        Else
            Select Case YChoice
                Case Executable_Time
                    Graph1.GraphData = ThesisRec.EXTIME
                Case Time_Util
                    Graph1.GraphData = ThesisRec.TIMEUTIL
                Case Storage
                    Graph1.GraphData = ThesisRec.STOR
                Case Mem_Util
                    Graph1.GraphData = ThesisRec.MEMUTIL
                Case Pi_Mult
                    Graph1.GraphData = ThesisRec.PI
                Case Analyst_Experience
                    Graph1.GraphData = ThesisRec.AEXP
                Case Lang_Experience
                    Graph1.GraphData = ThesisRec.LEXP
                Case Actual_KDSI
                    Graph1.GraphData = ThesisRec.ACTKDSI
                Case Equivalent_DSI
                    Graph1.GraphData = ThesisRec.EDSI
                Case Actual_Effort
                    Graph1.GraphData = ThesisRec.ACTEFFORT
            End Select
        End If
        'Debug.Print Str(Graph1.GraphData)
```

```
        more = AgiViewNext(DBHandle)
    Loop
    Screen.MousePointer = Default
End Sub
```

Code for ZoomData Form

```
Option Explicit
Dim TempYMax As Single
Dim TempYMin As Single
Dim TempXMax As Single
Dim TempXMin As Single

Sub cmdCancel_Click ()
    Unload frmZoomData
End Sub

Sub cmdOK_Click ()
    YMax = Val(txtYMax.Text)
    YMin = Val(txtYMin.Text)
    XMax = Val(txtXMax.Text)
    XMin = Val(txtXMin.Text)

    Unload frmZoomData
End Sub

Sub cmdReset_Click ()
    txtYMax.Text = Str(TempYMax)
    txtYMin.Text = Str(TempYMin)
    txtXMax.Text = Str(TempXMax)
    txtXMin.Text = Str(TempXMin)

    txtYMax.SetFocus
End Sub

Sub Form_Load ()
    txtYMax.Text = Str(YMax)
    txtYMin.Text = Str(YMin)
    txtXMax.Text = Str(XMax)
    txtXMin.Text = Str(XMin)

    TempYMax = YMax
    TempYMin = YMin
    TempXMax = XMax
    TempXMin = XMin
End Sub
```

Appendix D

Block Change Process Models

Table of Contents

Object Oriented Design Models

Data Dictionary

Data Dictionary Entry Format and Definitions

Entity Label := **Entity Role: Definition**

where

Entity Label is the name of the entity appearing in the diagram.

":=" is "defined as"

Entity Role is the role the entity plays in the diagram. Roles are either

Objects, Processes, Attributes, Data, or *Relationships.*

Definition is the definition of the entity within the context of the model.

a) *Objects* are physical or abstract actors that have meaning in the problem space.
b) *Objects* are characterized by *Attributes* and are linked together by *Relationships.*
c) *Attributes* are features of an *Object* that have meaning in the problem space and need to be maintained in the model space.
d) *Relationships* describe how one *Object* interacts with another.
e) *Data* is information used or created by a *Process.*

Approved ECP := Object: Engineering Change Proposal that has been reviewed and approved by the Configuration Control Board and is to be included in the current block change cycle

Approves := Relationship: "the *CCB* approves the *ECP*"

Availability := Attribute: describes how much time computer assets owned by the *SSA* can be used during the development phase of the current block change cycle

Block Change := Object: all materials produced by the *SSA* during the block change process that are released to the field

CCB := Object: formal group (Configuration Control Board) comprised of *User* personnel and *SSA* personnel responsible for reviewing and approving all changes to be made to the *Product Baseline* during the current block change cycle

CCB Process := Process: actions taken by the *CCB* to review and approve an *ECP* for inclusion in the current block change

Change/Problem Report := Object: standard forms that describe, in terms of requirements, what needs to be added, deleted, or modified in the *Product Baseline*

Change Requests := Data: standard form that describes, in terms of requirements, what needs to be added, deleted, or modified in the *Product Baseline*

Code := Object: the executable component of *Software*

Complexity := Attribute: describes the design/structure of the including the completeness and readability of the available documentation, readability of the available code listings, and the degree of coupling and cohesion within the *Code* itself

Cost := Attribute: estimate in dollars of the *Resources* needed to complete the ECP

Data Size := Attribute: size of the database/data structures required by the *Code* to perform it's function according to the *Requirements*

Demonstrates := Relationship: "the *Test Plan* demonstrates the *Block Change* (meets the user's requirements)"

Depot MX := Object: Depot maintenance *Manuals* for the weapon system

Description := Attribute: the portion of the *Change/Problem Report* that describes what is wrong or what needs to be done to the software which forms the requirement for the next block change cycle

Design Docs := Object: documents (generally delivered with the software) that describe the software design at various levels of abstraction

Develop Changes := Process: translation by the *SSA* of the approved *ECP* into executable software for *Ground Integration Test*

Development Paradigm := Attribute: the specific engineering management method used to plan and track the *Develop Changes* process

Draft ECP := Data: *ECP* prior to approval by the *CCB*

ECP := Object: collection of one or more *Change Problem Reports* that have been approved for inclusion into the *Product Baseline* during the next block change cycle

Efficiency := Attribute: the average *SSA* manhours of effort it takes to change a single line of code measured from the *Effort Estimation* process to the delivery of a test tape to the *Ground Integration Test* process

Equipment := Object: Computer hardware, software, and tool sets owned by the *SSA* that can be applied to the current block change cycle

Estimate Effort := Process: comprises investigating, prioritizing, and estimating the effort required to implement each *Problem Report* and *Change Request* waiting to be implemented in the current block change cycle

Effort Estimate := Data: documented result of the *Effort Estimate* process

Experience := Attribute: describes *SSA* programmer and analyst personnel experience with the software being modified and with the *SSA Equipment*

Field Change := Process: distributing the *Updated Tapes &Manuals* to the *User*

Field MX := Object: current configuration manuals used by the *Field MX Personnel*

Field MX Personnel := Object: weapon system maintainers that are assigned to the weapon system's base of operations

Fielded to := Relationship: "the *Block Change* is fielded to the *User*"

Fliers := Object: weapon system operators

Flight Crew := Object: current configuration manuals used by *Fliers*

Flight Failed Tape := Data: *Software* with major problems that were uncovered during *Flight Test* requiring *Rework*

Flight Problems Not Waivered := Data: *Software* with minor problems uncovered during *Flight Test* that requires *Rework*

Flight Test := Process: software testing that occurs on one or more test aircraft in support of development testing or operational testing

Flight Test Equipment := Object: the aircraft to be used during *Flight Test*

Flight Test Plan := Object: documentation that describes the objectives of the *Flight Test* process in terms of *Flight Test Procedures*, *Equipment*, and *Results*

Flight Test Procedures := Object: documentation describing the mission profiles, and series of actions to be used during the *Flight Test* process

Flight Test Results := Object: documentation describing the outcome of the *Flight Test* process

FT Schedule := Attribute: description of the availability of *Flight Test Equipment*

Ground Integration Test := Process: software ground testing done in the avionics system integration facility or other system mockup

Ground Test Equipment := Object: hardware and facilities used during *Ground Integration Test*

Ground Test Plan := Object: documentation describing the objectives of the *Ground Integration Test* process in terms of procedures, equipment, and results

Ground Failed Tape := Data: software with major problems that were uncovered during *Ground Integration Test* requiring *Rework*

GT Speed := Attribute: description of the speed of the host computer used during *Ground Integration Testing*

GT Memory := Attribute: description of the amount of unused memory in the host computer used during *Ground Integration Testing*

GT Procedures := Object: documentation describing the series of actions to be accomplished during the *Ground Integration Test* process

GT Results := Object: documentation describing the outcome of the *Ground Integration Test* process

Integrated Waivered Tape := Data: *Software* with minor problems discovered during the *Ground Integration Test* process that is being released to the *Flight Test* process

Integrated Tape := Data: *Software* with no problems discovered during the *Ground Integration Test* process that is being released to the *Flight Test* process

Integration Problems Not Waivered := Data: *Software* with minor problems discovered during the *Ground Integration Test* process that requires *Rework*

Local Policy := Object: documentation that describes the *SSA* software *Development Paradigm*, and *Standards*

Manuals := Object: family of documents describing the weapon system operation and maintenance practices and procedures

Memory := Attribute: description of the unused portion of the *Target Computer* RAM and ROM

Minor Flight Problems := Data: *Software* with minor problems uncovered during the *Flight Test* process

Minor Integration Problems := Data: *Software* with minor problems uncovered during the *Ground Integration Test* process

Operational Tape := Data: current weapon system *Software* and supporting *Manuals*

Personnel := Object: people employed by the *SSA*

Prioritize Board := Process: rank ordering of changes to be made to the *Software* and preparation of an *ECP* for those changes

Priority := Attribute: describes the urgency of the *Change/Problem Report*

Problem Reports := Data: description of problems or errors in the current *Product Baseline* that must be corrected through additions, deletions, or modifications

Procedures := Object: all previous documented test actions dating from development to the current block change cycle

Produces := Relationship: "the *SSA* produces the *Block Change*"

Product Baseline := Object: documentation that describes the configuration of *Software*, *Manuals*, and *Tests* that exist at the start of a block change cycle

Reliability := Attribute: describes the required reliability of the baseline *Software*

Reproduce Tape := Process: copy sufficient quantities of *Block Change* materials to accomplish the *Field Change* process

Requirements := Object: documented *User* needs that the *Software* must contain

Resides in := Relationship: "the *Block Change* resides in the *Target Computer*"

Resources := Attribute: describes the *SSA Equipment*, *SSA Personnel*, *Flight Test Equipment*, and *Ground Test Equipment* needed to accomplish the work described in the *ECP*

Results := Object: documentation describing the outcome of all previous *Procedures*

Revised Tapes := Object: new *Product Baseline Software*

Revised Manuals := Object: new *Product Baseline Manuals*

Rework := Process: activity that corrects problems found during *Ground Integration Test* and/or *Flight Test*

Schedule := Attribute: describes the expected completion dates of major *ECP* activities

Size := Attribute: describes the number of lines of code *Product Baseline Software* contains

Size Estimate := Attribute: describes the estimated number lines of code that must be added, deleted, or modified to satisfy the *Change/Problem Report*

Skill := Attribute: describes the relative abilities of the *SSA Personnel* working on the block change

Software := Object: *Requirements*, *Design Docs*, and *Code*

Speed := Attribute: describes the throughput of the *Target Computer*

SSA := Object: Software Support Agency, the organization responsible for *Software* support

Standards := Object: documentation that describes the *SSA*'s local *Software* development policies

Starts From := Relationship: "the *Block Change* starts from the *Product Baseline*"

Submits := Relationship: "the *User* submits a *Change/Problem Report*"

Target Computer := Object: the computer that resides on the weapon system

Tests := Object: all previous *Procedures*, *Test Code*, and *Results* dating from development to the current block change cycle

Test Assets := Attribute: describes what equipment will be required to support the *Develop Changes*, *Ground Integration Test*, and *Flight Test* processes for a particular *Change/Problem Report*

Test Code := Object: the current configuration of *Software* created to support the *Develop Changes* and *Ground Integration Test* processes

Test Plan := Object: documentation that describes how, why, where, and when the *Software* will be tested

Test Tape := Data: *Software* containing the changes described in the *ECP* that has completed unit testing

Test Schedule := Attribute: describes when all portions of the *Test Plan* are scheduled to begin and end

Updated Tapes &Manuals := Data: sufficient quantities of all *Block Change* materials necessary for distribution to all weapon system bases of operation

User := Object: weapon system operators and maintainers located at the weapon system site

Waiver Board := Process: formal organization that reviews minor *Software* problems discovered during testing responsible for returning the *Software* for *Rework* or passing it through to the next process and generating a *Problem Report* to be acted on during the next block change cycle

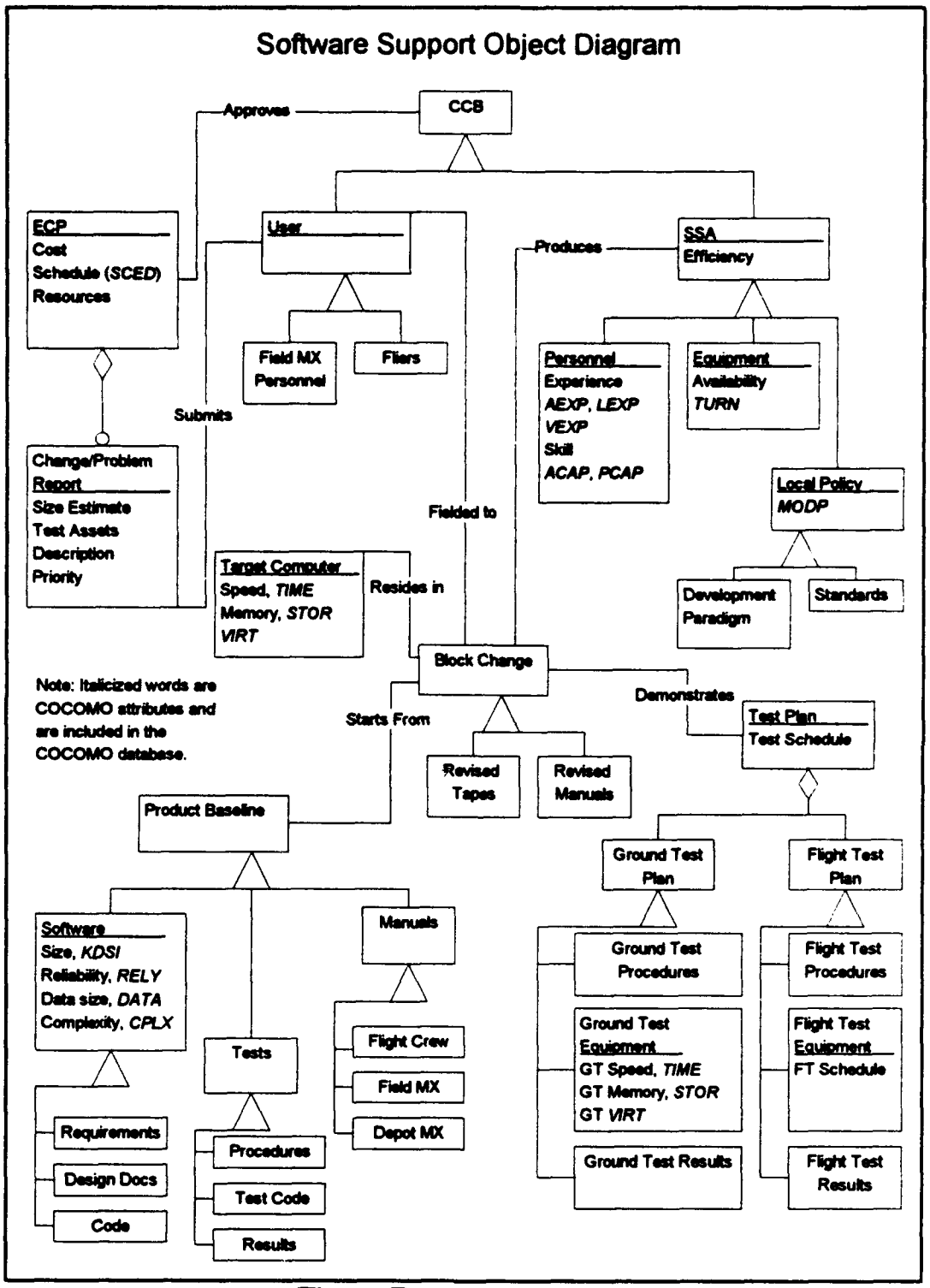


Figure D.1 - Object Model

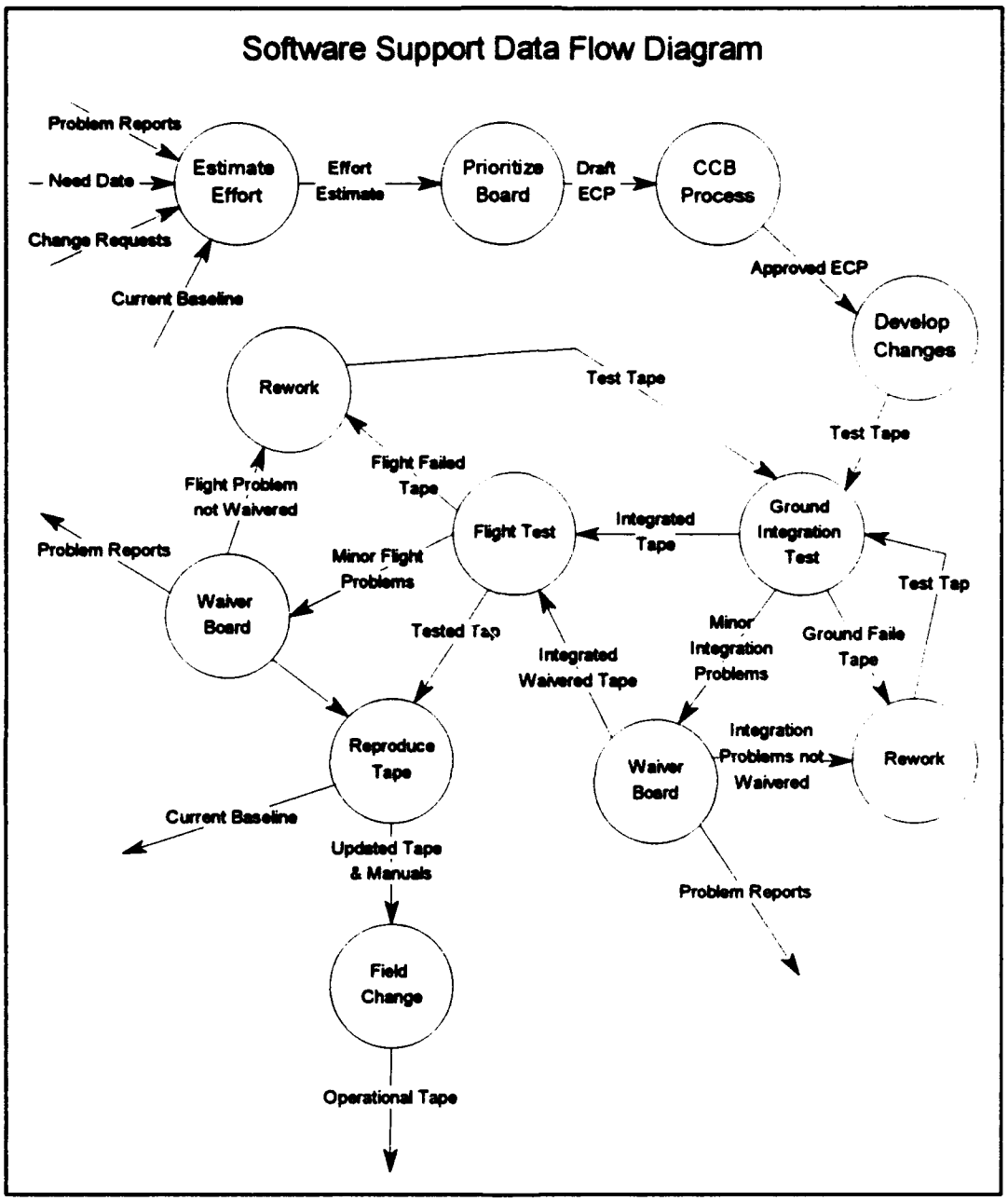


Figure D.2 - Functional Model For Single Block Change Cycle

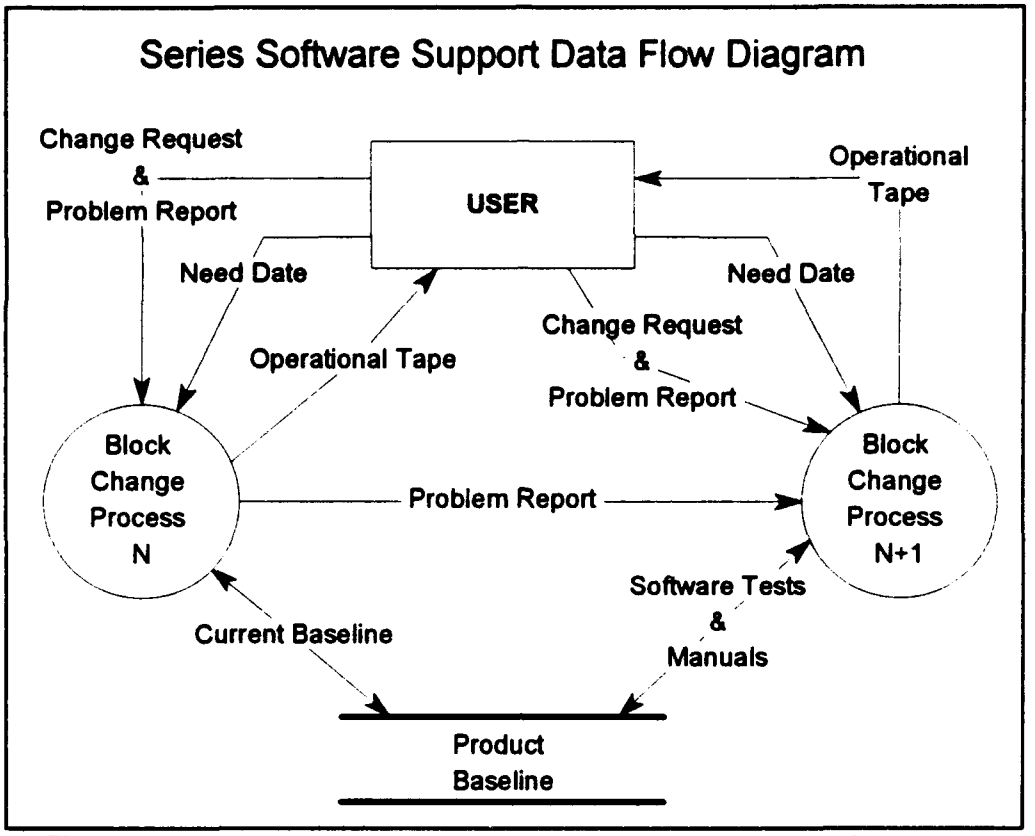


Figure D.3 - Functional Model For Block Change Cycle Series

Sacramento ALC Block Change Process

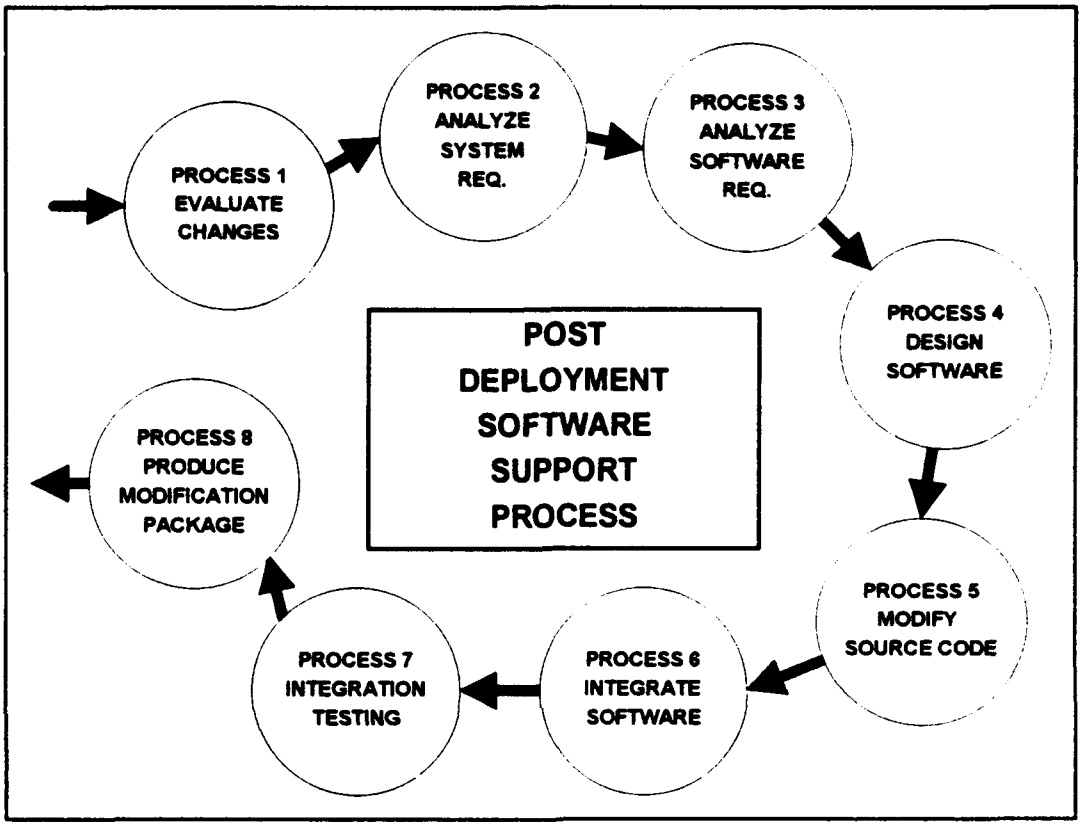


Figure D.4 - Sacramento ALC Block Change Process

MIL-HDBK-347 Block Change Process

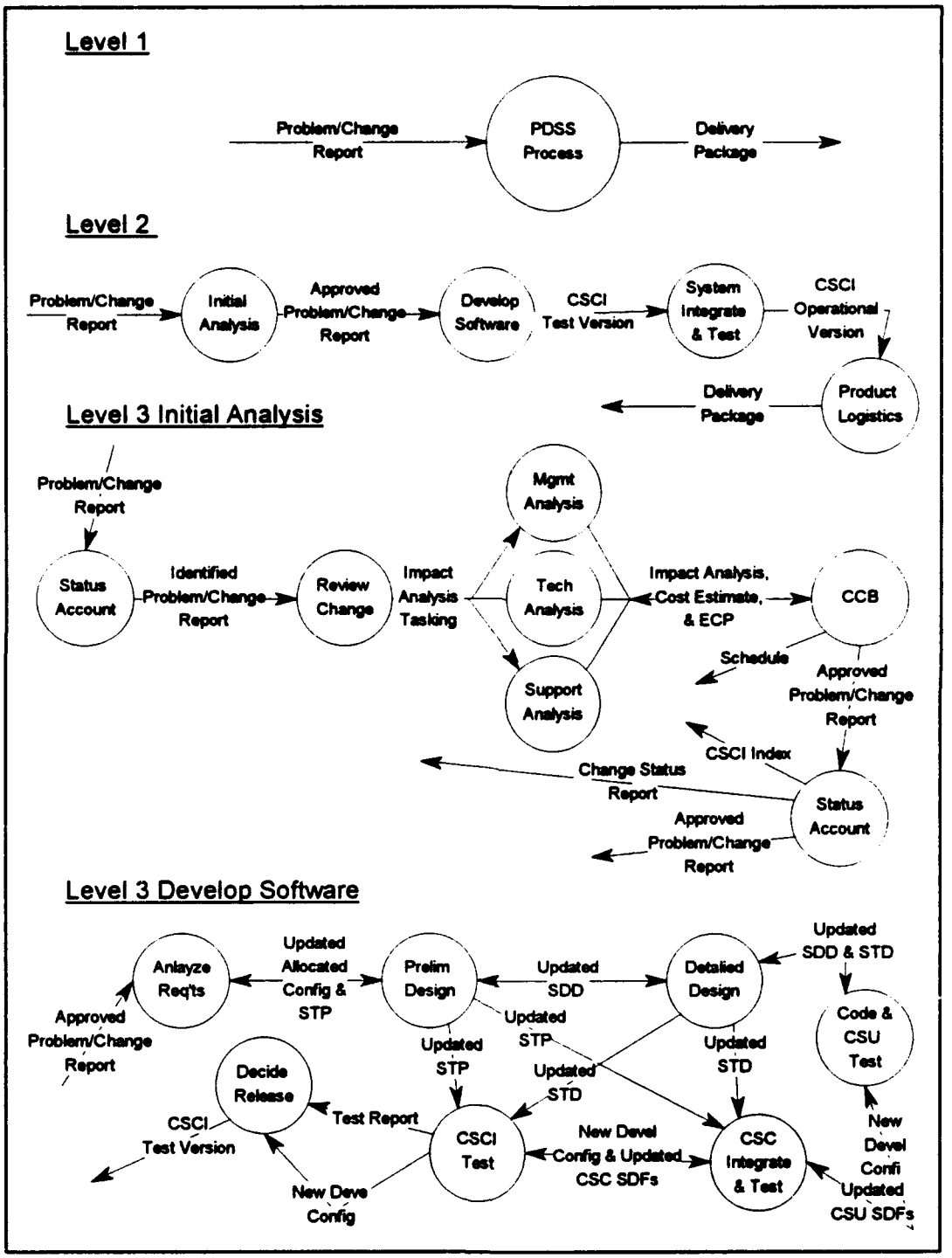


Figure D.5a - MIL-HDBK-347 Block Change Process

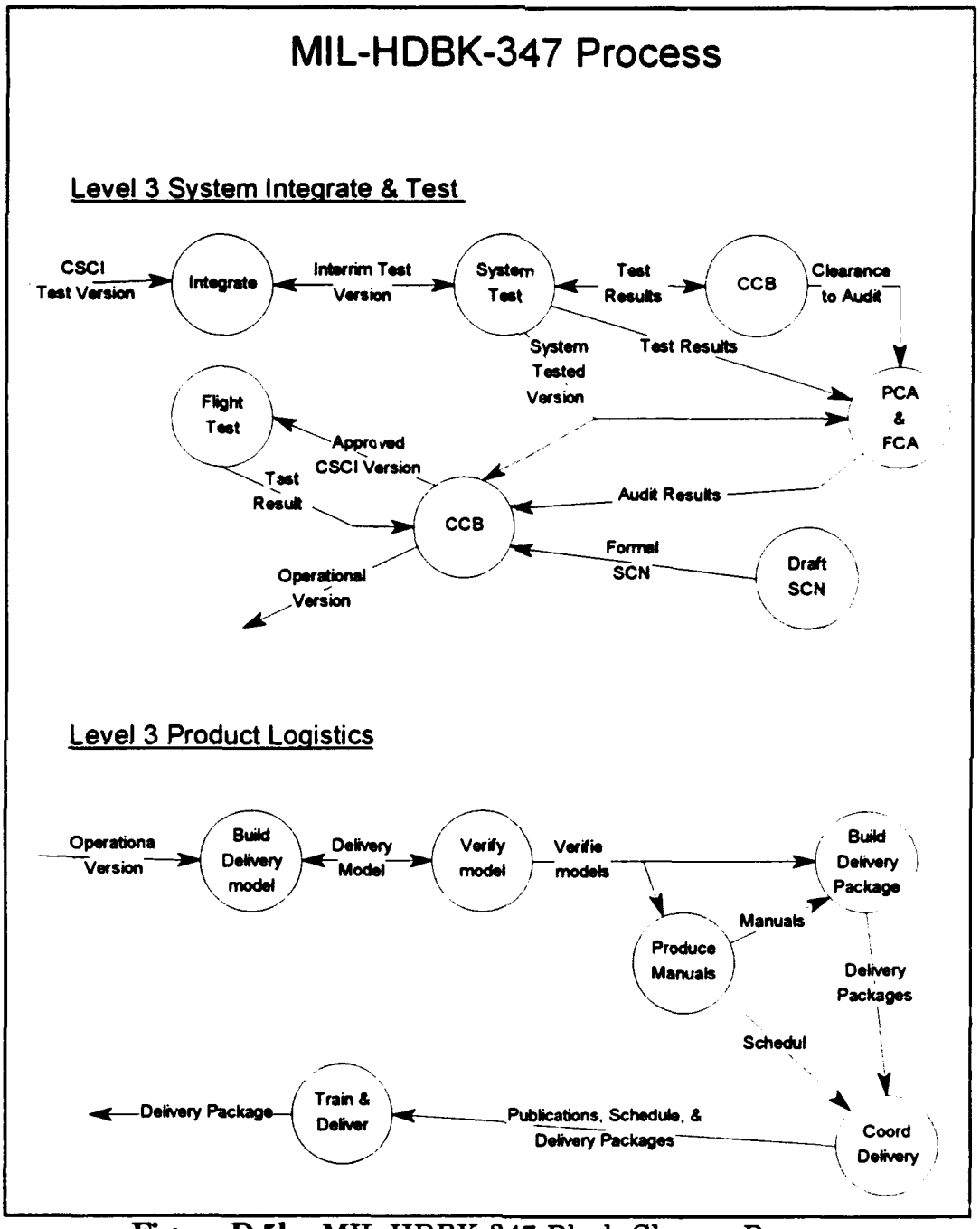


Figure D.5b - MIL-HDBK-347 Block Change Process

Appendix E

Memory and Throughput Relationship Derivation

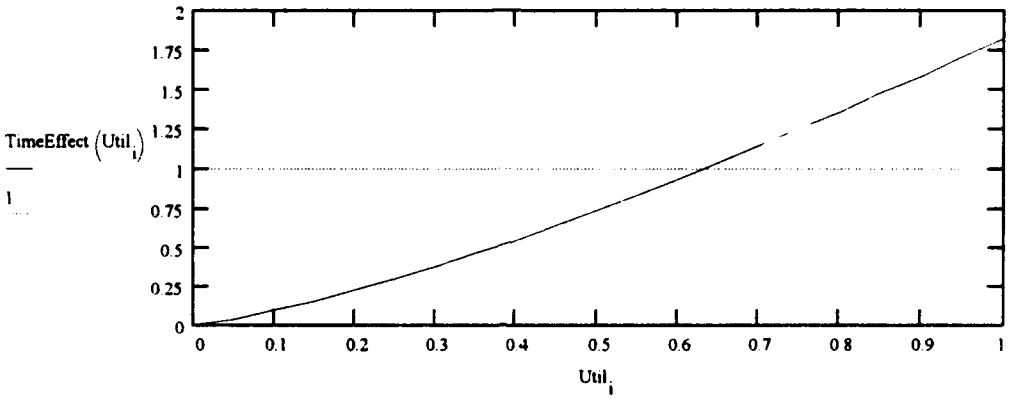
This derivation was performed on MathCad, version 4.0

Start with the SYSCON equations from SYSCON Report (pg 41) or Table 2.3.
Note these equations can be calibrated separately from the coefficient and exponent if given the proper historical data. Find:

% Timing Fill, result of averaging Design & Development Coefficients and Exponents

$$Coeff = \frac{1.82 + 1.82}{2} \qquad Coeff = 1.82 \qquad Exp = \frac{1.30 + 1.31}{2} \qquad Exp = 1.305$$

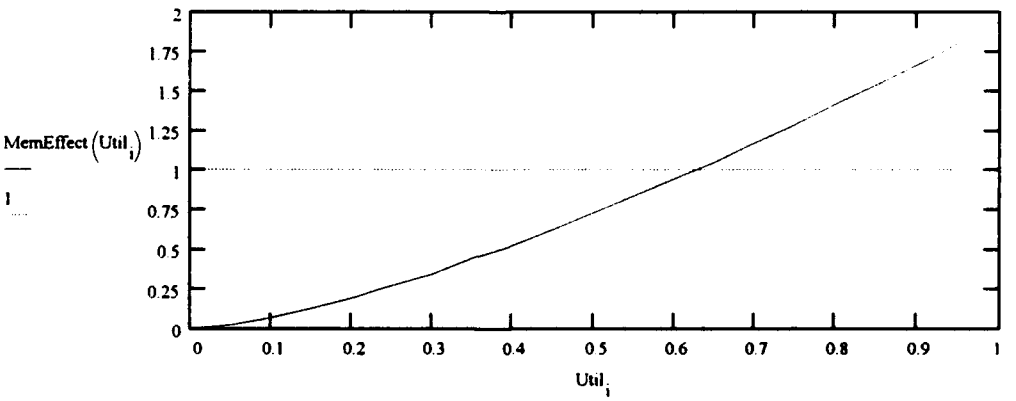
$$TimeEffect(X) = Coeff \cdot X^{Exp} \qquad N = 20 \qquad i = 0..N \qquad Util_i = \frac{i}{N}$$



% Memory Fill, result of averaging Design & Development Coefficients and Exponents

$$Coeff = \frac{2.00 + 1.88}{2} \qquad Coeff = 1.94 \qquad Exp = \frac{1.50 + 1.35}{2} \qquad Exp = 1.425$$

$$MemEffect(X) = Coeff \cdot X^{Exp} \qquad N = 20 \qquad i = 0..N \qquad Util_i = \frac{i}{N}$$



Maximum total contribution is

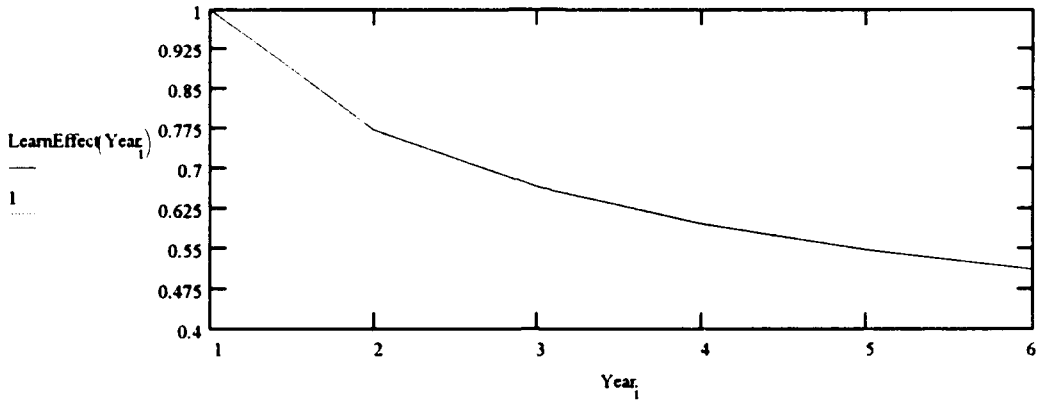
$TimeEffect(1) = 1.82$ $MemEffect(1) = 1.94$ $TimeEffect(1) \cdot MemEffect(1) = 3.531$

$TimeEffect(.95) = 1.702$ $MemEffect(.95) = 1.803$ $TimeEffect(.95) \cdot MemEffect(.95) = 3.069$

Learning Effect Table 2.5, is the result of averaging the Design & Development Coefficients and Exponents from the SYSCON relationship

$$Coeff = \frac{1.64 + 1.65}{2} \quad Coeff = 1.645 \qquad Exp = \frac{-.375 + -.374}{2} \quad Exp = -0.375$$

$$LearnEffect(X) = \frac{Coeff \cdot X^{Exp}}{Coeff} \qquad N = 6 \qquad i = 1..N \qquad Year_i = i$$



Appendix F

Database Contents

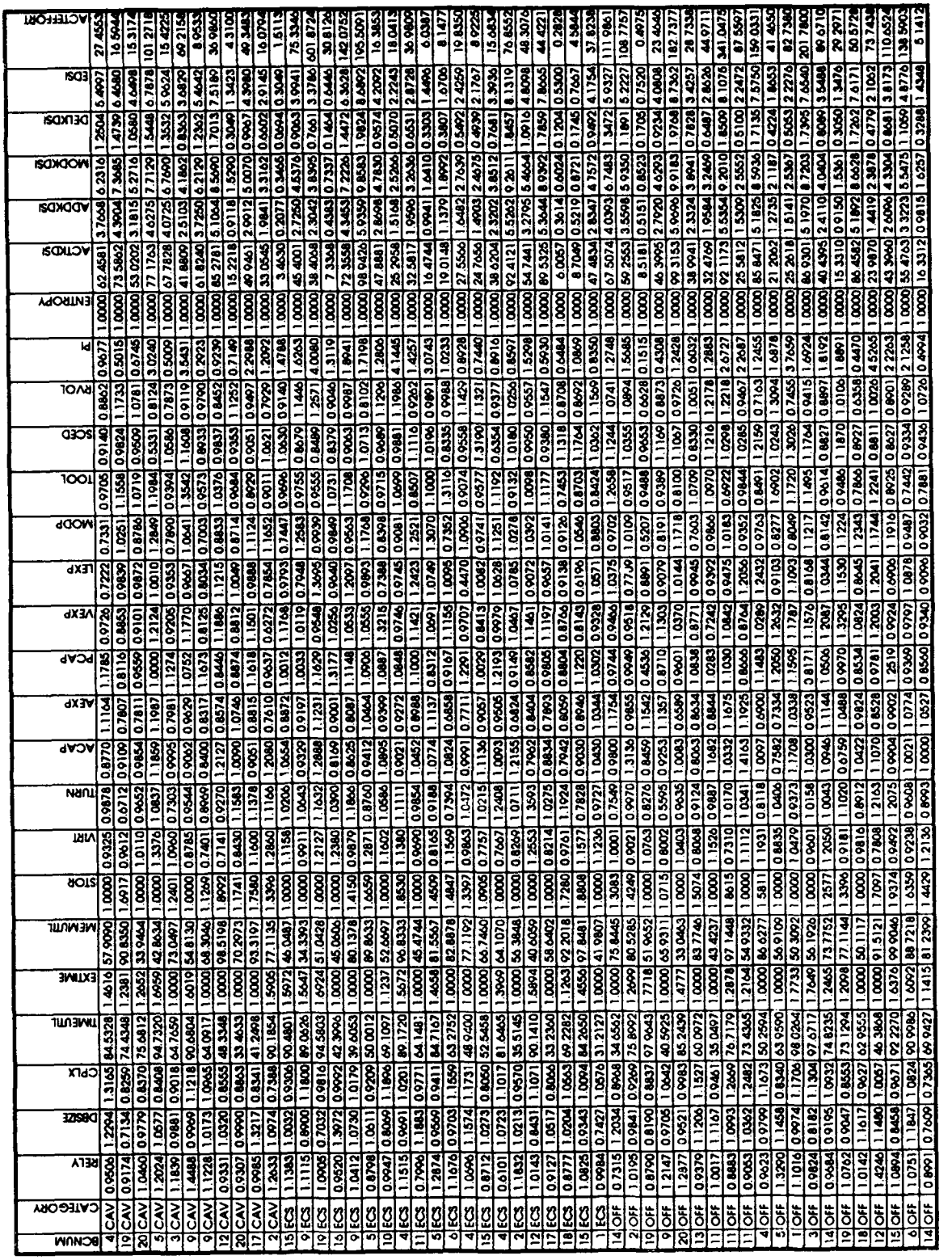
Composite Database

Part A

RCNUM	CATEGORY	RELV	DBSIZE	CPLX	TIMEUTIL	EXTIME	STOR	MEMUTIL	VIRT	TURN	ACAP	AEXP	PCAP	VEXP	LEXP	MODP	TOOL	SCED	RVOL	PI	ENTROPY	ACTDOI	ADDKDSI	MODKDSI	DELKDSI	EDSI	ACTEFFORT
3	CID	0.9825	1.0794	1.0653	86.1986	1.4993	86.9203	1.5887	0.9480	0.9938	1.1761	1.1618	1.3011	0.8852	0.8452	0.7389	1.0694	0.9418	0.9705	1.8811	1.0000	5.0310	0.3016	0.5058	0.1006	0.4440	1.1606
13	CID	0.6417	0.9378	1.0261	60.7058	1.0000	64.3118	1.0000	0.6719	0.8615	0.8347	1.0193	1.2262	1.0691	1.1381	1.0706	0.8305	1.0411	0.9492	0.3986	1.0000	95.9733	5.7679	9.2160	1.9132	8.4586	10.2512
16	CID	1.0874	0.8732	1.0824	85.4882	1.4832	80.0967	1.4140	1.3949	1.2157	0.7597	0.9897	0.7697	0.8538	1.1173	1.2066	0.9502	0.8159	1.2879	2.5068	1.0000	68.6492	4.1132	6.8663	1.3682	6.0375	42.1934
13	CID	0.6375	0.8271	1.0859	82.7868	1.4224	52.5514	1.0000	1.2325	1.3020	0.7493	1.4036	1.0314	1.0202	1.0024	0.9103	1.0356	1.2204	0.9628	1.7020	1.0000	29.1624	1.7500	2.9095	0.5808	2.5627	18.1664
7	CID	1.2051	1.1478	1.1370	76.5617	1.2844	84.7960	1.5337	1.1240	1.2450	1.0285	1.3634	0.7929	1.0782	0.8603	1.0139	0.6277	1.2160	0.7790	2.6964	1.0000	78.9288	4.7039	7.8785	1.5819	6.9224	52.5163
12	CID	1.1012	0.8405	1.2701	87.4916	1.5288	70.7943	1.4004	1.2648	1.1024	0.9020	0.6538	0.9451	1.0779	1.1128	1.3886	1.1116	1.0092	1.0119	3.4722	1.0000	64.0769	3.8810	6.4544	1.2936	5.6850	57.6885
9	CID	1.1078	0.9536	0.9908	97.9740	1.7720	55.3540	1.0000	1.0688	1.2274	0.9069	0.7408	0.8832	0.7409	0.9190	1.2374	0.9630	0.9591	0.7040	0.7752	1.0000	28.4478	1.7113	2.8499	0.5692	2.5085	5.4113
9	CID	1.0806	0.9022	1.2000	87.1251	1.5204	39.1204	1.0000	1.1001	0.9197	1.0226	1.2011	1.0132	1.0544	0.9650	1.3190	1.0157	1.1248	0.9511	3.2601	1.0000	57.5950	3.4508	5.7649	1.1561	5.0706	46.8187
2	CID	0.8410	1.1051	0.9959	65.7798	1.0539	33.6067	1.0000	1.1868	0.9795	1.0556	1.0545	1.0019	1.1527	0.8695	1.2525	1.2283	1.1526	1.1317	2.5433	1.0000	34.7845	2.0840	3.4623	0.6959	3.0512	18.3256
1	CID	0.7908	1.1760	0.8811	41.4918	1.0000	51.1337	1.0000	0.8434	0.8850	1.1684	0.6093	1.0419	0.9079	0.9183	1.1869	1.2405	0.7558	1.0491	0.4416	1.0000	87.9948	5.3005	8.7948	1.7633	7.7504	9.7413
12	CID	1.0997	0.9098	1.0654	53.5336	1.0000	78.8900	1.3838	1.2922	1.0768	0.7894	0.9297	1.1153	1.1030	0.9652	0.9935	1.1397	0.8443	0.9303	1.6955	1.0000	40.5169	2.4322	4.0455	0.8096	3.5627	17.1740
19	CID	1.0232	0.5971	0.7406	67.8665	1.0981	43.9021	1.0779	1.0480	0.9916	1.0683	0.7865	1.1003	1.1478	0.9945	0.9308	1.2061	0.6678	0.2253	1.0000	60.9544	3.6507	6.1063	1.2233	5.3672	3.6109
7	CID	0.6685	1.0063	0.8746	82.2903	1.4112	58.0399	1.0000	1.1782	1.1651	1.2004	1.0173	0.9736	0.9085	0.9945	1.0846	0.8591	1.1797	1.0259	1.4754	1.0000	13.2233	0.7963	1.3100	0.2643	1.1599	2.6578
14	CID	1.2123	0.9784	1.3148	91.1438	1.6126	33.0175	1.0000	1.0373	1.2408	1.0329	1.2334	0.8456	1.3016	1.1182	0.6752	0.9177	1.0191	0.7165	1.7497	1.0000	6.5520	0.3922	0.6556	0.1307	0.5762	2.7582
1	CID	0.6921	1.1625	0.9104	50.6044	1.0000	65.3924	1.0591	0.8751	0.6531	0.7369	0.8441	1.2266	1.2208	0.8439	1.3178	1.0946	1.2756	1.2533	0.8109	1.0000	45.4816	2.7180	4.5550	0.9069	3.9994	11.2617
3	CID	0.9812	1.1340	1.0757	53.5385	1.0000	51.5659	1.0000	1.3765	1.0986	1.1047	1.2072	0.9095	1.2446	0.9719	1.0385	0.9950	1.0743	1.2473	2.0167	1.0000	54.8700	3.3141	5.5010	1.0980	4.8475	21.6998
15	CID	0.8638	0.8283	0.7949	84.5308	1.4616	72.7833	1.2337	0.8800	0.9789	1.1530	1.2478	0.9295	1.2446	0.9719	1.0385	0.9950	1.0743	1.2473	2.0167	1.0000	54.8700	3.3141	5.5010	1.0980	4.8475	26.2777
7	CID	1.0755	1.1010	1.1493	54.8851	1.0000	80.1333	1.4149	1.2157	1.0902	1.1442	0.7854	1.0857	1.1786	1.1013	0.9274	0.9420	1.0466	1.0381	3.1224	1.0000	60.9734	3.6491	6.1020	1.2215	5.3939	51.3280
12	CID	1.2631	0.9114	0.9847	47.0240	1.0000	69.5485	1.1563	1.0536	1.3382	0.9170	1.2842	0.7383	1.1287	1.1666	1.0026	0.8866	0.9987	1.0957	2.0946	1.0000	83.9667	5.0397	8.4245	1.6804	7.4044	44.7675
15	CID	0.6401	1.2091	0.8685	72.4886	1.1940	46.8055	1.0000	0.9567	1.0250	1.0026	1.0027	0.9603	1.2530	1.0023	1.3920	1.0171	0.9073	1.2744	1.7626	1.0000	54.9535	3.3054	5.5153	1.1017	4.8516	28.6434
3	NAV	1.1922	0.6108	0.9400	30.9388	1.0000	62.5242	1.0000	1.0068	1.0228	1.0350	0.9423	1.2346	0.7468	0.7764	1.3720	1.1629	0.8192	1.0342	0.6652	1.0000	0.3498	0.0210	0.0349	0.0070	0.0307	0.2074
17	NAV	1.0828	0.9782	1.0460	31.1534	1.0000	37.1685	1.0000	1.2159	1.2964	1.2133	1.1656	0.7904	0.9047	0.9909	1.0799	1.0317	0.9921	0.9959	1.9209	1.0000	91.8841	5.5207	9.2440	1.8381	8.1183	92.4483
11	NAV	1.3095	1.0578	0.8977	73.0298	1.2076	76.0460	1.3132	1.0421	0.8930	1.0816	1.1148	0.6974	0.9803	0.9085	0.8851	1.0803	0.9221	0.8628	1.0454	1.0000	96.5974	5.8120	9.6026	1.9344	8.4805	49.1518
6	NAV	0.8783	1.0602	0.7905	83.1843	1.4313	84.3082	1.5211	0.7954	0.9773	1.2031	1.0128	1.1167	1.0185	1.1086	0.9163	0.9779	1.3431	0.7700	1.7736	1.0000	26.8432	1.6086	2.6821	0.5360	2.3598	19.9705
6	NAV	0.9699	0.8324	0.9853	89.9517	1.5851	81.9501	1.4609	0.9806	0.9654	0.8397	1.2615	1.2979	0.8617	0.9107	0.9888	0.9411	1.0323	1.2102	1.6027	1.0000	90.2870	5.4092	9.0329	1.7979	7.9411	75.6926
18	NAV	0.8131	0.8176	1.1357	48.3859	1.0000	98.8106	1.9072	1.0463	1.2849	1.1711	1.2543	0.8390	1.0865	1.2590	0.8474	1.1932	1.0477	1.2915	4.4511	1.0000	39.9090	2.3910	4.0026	0.7991	3.5163	78.1664
2	NAV	0.9746	1.0008	0.5484	85.2593	1.4781	69.5723	1.1568	1.0381	1.1316	0.8434	1.0334	1.1484	1.2347	0.9481	0.9265	0.8111	1.0723	1.0317	1.0464	1.0000	23.9985	1.4450	2.4014	0.4810	2.1146	11.7727
6	NAV	0.6451	1.0224	1.0270	44.0031	1.0000	78.7616	1.3805	1.0350	1.3004	0.9911	0.8707	1.0402	0.9072	0.9364	1.2931	1.3023	0.9254	1.0015	1.5049	1.0000	36.1997	2.1700	3.6130	0.7258	3.1817	25.0040
20	NAV	1.0657	1.0322	1.0645	80.0072	1.3604	45.7486	1.0000	0.8921	1.2236	1.2588	1.0458	1.0283	1.2323	1.0786	1.1315	1.0771	1.3225	1.0657	5.4755	1.0000	65.3084	3.9384	6.5598	1.3008	5.7557	167.7160
6	NAV	1.0151	1.1151	0.6928	65.0204	1.0378	39.7472	1.0000	0.9996	1.0022	0.8829	0.9124	1.1075	1.2217	0.5398	1.0663	0.8871	0.9660	0.9265	0.3974	1.0000	45.6732	2.7394	4.5603	0.9130	4.0147	9.9973
12	NAV	1.2003	0.9804	1.0775	42.7501	1.0000	34.8149	1.0000	0.8851	0.8156	1.0450	1.0930	1.1500	0.8672	1.0465	0.8287	0.9097	0.7313	1.0363	0.6267	1.0000	6.7952	0.4087	0.6810	0.1357	0.5995	1.4788
16	NAV	1.1043	0.8630	0.9667	53.5625	1.0000	84.8532	1.5352	1.1094	1.0818	1.2714	1.0173	0.6998	1.2955	1.1285	0.8006	0.9483	1.2952	1.1991	2.6788	1.0000	23.8279	1.4039	2.3566	0.4716	2.0682	24.3480
18	NAV	0.8704	1.1039	0.9348	99.5426	1.8091	78.6072	1.3767	0.8271	0.9015	0.8203	0.6427	1.0031	0.9340	0.7620	0.6409	0.9010	0.9927	1.0109	0.3864	1.0000	66.0372	3.9843	6.5799	1.3168	5.8003	12.9799
7	NAV	1.2745	1.2635	1.1977	43.0395	1.0000	90.4613	1.8429	0.9457	0.8560	1.4460	1.0537	1.0412	0.9468	1.0665	0.8249	1.2707	0.9317	1.0216	4.7837	1.0000	85.6124	5.1261	8.5634	1.7036	7.5271	183.1132
17	NAV	0.7323	0.9213	0.6077	70.1200	1.1452	87.2705	1.5079	0.9126	1.1707	1.1007	1.0126	1.0834	1.0954	0.9373	1.0594	1.0105	1.1301	1.1155	1.3643	1.0000	29.5728	1.7670	2.9498	0.5917	2.6050	19.5912
2	NAV	0.8677	0.8040	0.6899	36.9965	1.0000	95.2133	1.8090	0.9481	1.2502	1.0274	0.9974	1.2652	1.0870	0.8228	1.3163	1.0353	1.0782	0.8529	1.5009	1.0000	33.0990	1.9797	3.3066	0.6614	2.9076	17.3372
7	NAV	0.9051	1.1012	0.9427	70.2448	1.1479	58.2536	1.0000	0.7840	1.0408	1.0824	1.1500	0.9371	1.1373	0.7941	1.1590	1.0378	0.8355	0.7454	0.6929	1.0000	92.9104	5.5558	9.2563	1.8550	8.1499	33.3099
1	NAV	0.7882	0.9400	0.8078	47.2652	1.0000	99.1990	1.9179	1.2658	0.9158	1.0990	0.9786	0.9630	0.8570	0.9172	0.8603	0.9732	0.9607	0.8239	0.7196	1.0000	1.0751	0.0644	0.1073	0.0216	0.0945	0.0176
5	NAV	0.9364	0.9733	0.9168	52.1731	1.0000	70.7009	1.1637	0.9690	1.1974	1.2292	1.0356	1.2456	1.1860	1.3829	0.9263	1.1749	1.0847	0.7982	2.8122	1.0000	94.9195	5.6867	9.4317	1.8925	8.3172	128.1715
2	CAV	0.9828	1.2729	0.9046	60.8034	1.0000	80.2545	1.4180	1.2294	1.1161	0.9152	0.9860	0.8605	0.8443	0.9853	1.2949	0.9430	1.1637	1.0032	2.0713	1.0000	7.1188	0.4267	0.7182	0.1419	0.6278	1.6507
11	CAV	0.7910	1.0804	0.9193	45.7098	1.0000	35.1985	1.0000	0.8320	1.0363	1.0867	0.8319	0.8777	1.0340	1.2183	0.9129	0.8224	1.0659	1.2391	0.5086	1.0000	15.6514	0.9407	1.5653	0.3121	1.3780	2.7540
11	CAV	0.8258	0.7369	1.0156	81.1047	1.3861	97.7155	1.8772	0.6287	1.1267	0.8094	0.9417	0.8686	1.1476	1.0216	0.9419	1.2990	0.8363	0.9264	0.8382	1.0000	93.8083	5.6211	9.4130	1.8775	8.2679	38.8087
14	CAV	1.2328	1.2043	0.9414	98.1594	1.7764	89.8321	1.6651	1.2780	1.1049	1.2637	1.0922	1.2248	1.2458	1.4120	0.8497	0.9393	0.8768	0.6559	6.6674	1.0000	60.1930	3.5889	6.0078	1.2051	5.2794	203.6187
16	CAV	0.9661	0.9459	0.9244	59.4038	1.0000	88.4182	1.6279	1.0570	1.2735	0.7450	1.6307	1.2364	0.8862	1.2012	0.8073	0.9043	0.8844	1.1818	2.2629	1.0000	28.7210	1.7242	2.8772	0.5770	2.5312	25.7600
5	CAV	1.2005	1.1328	1.0397	64.3266	1.0000	40.1878	1.0000	1.1172	1.1261	1.0707	0.9428	1.1486	1.2053	1.2134	0.9784	0.7004	1.1445	0.8925	2.3036	1.0000	58.0226	3.5110	5.8579	1.1711	5.1528	58.6709
12	CAV	1.0355	1.0442	0.9761	34.0754	1.0000	76.6984	1.3293	1.0484	0.7593	1.2337	0.8604	1.1459	1.0158	1.1756	0.7978	0.6413	1.1923	0.8361	0.8556	1.0000	4.4370	0.2670	0.4431	0.0882	0.3904	2.1038
1	CAV	1.1751	1.1584	0.9655	90.5162	1.7377	59.3514	1.0000	1.0682	1.4239	1.1380	0.9125	0.8802	0.8256	1.2858	0.7749	0.8684	1.3293	0.9747	2.9381	1.0000	66.0182	3.9975	6.6019	1.3137	5.8219	92.6930
18	CAV	1.1995	1.0542	0.8073	61.2341	1.0000	34.3920	1.0000	1.0365	0.8351	1.2601	1.1480	1.2848	0.9182	1.1262	0.8817	1.0016	1.0333	0.8915	1.3816	1.0000	38.5324	2.3096	3.8526	0.7757	3.3906	20.6644

Composite Database

Part B



BCNUM	CATEGORY	RELV	DBSIZE	CPLX	TIMEUTIL	EXTIME	MEMUTIL	STOR	VIRT	TURN	ACAP	AEXP	PCAP	VEXP	LEXP	MODP	TOOL	SCED	RVOL	PI	ENTROPY	ACTIDSI	ADDKDSI	MODKDSI	DELKDSI	EDSI	ACTEFFORT
4	CAV	0.9506	1.2294	1.3165	84.5328	1.4616	57.9090	1.0000	0.9325	0.9878	0.8770	1.1164	1.1785	0.9729	0.7222	0.7331	0.9705	0.9140	0.8862	0.9077	1.0000	62.4581	3.7668	6.2316	1.2604	5.4997	27.4553
19	CAV	0.9174	0.7134	0.8259	74.4348	1.2381	90.8350	1.6917	0.9612	0.6712	0.9109	0.7807	0.8116	0.8853	0.9839	1.0251	1.1558	0.9824	1.1733	0.5015	1.0000	73.5862	4.3904	7.3685	1.4739	6.4680	16.5944
20	CAV	1.0460	0.9779	0.8370	75.6812	1.2652	33.9464	1.0000	1.0110	0.9652	0.9854	0.7811	0.9659	0.9101	0.9872	0.8784	1.0719	0.9509	1.0781	0.6745	1.0000	53.0202	3.1815	5.2716	1.0580	4.6698	15.3174
5	CAV	1.2024	1.0577	0.8408	94.7320	1.6659	42.8634	1.0000	1.3376	1.0857	1.1859	1.1987	1.0000	1.2124	1.0010	1.2849	1.1984	0.5331	0.8124	3.0240	1.0000	77.1763	4.6225	7.7129	1.5448	6.7878	101.2718
3	CAV	1.1899	0.9981	0.9018	64.7659	1.0000	73.0497	1.2401	1.0960	0.7303	0.9995	0.7981	1.1274	0.9205	0.9353	0.7890	0.9394	1.0586	0.7873	0.5009	1.0000	67.7828	4.0725	6.7690	1.3532	5.9624	15.4225
9	CAV	1.4488	0.9969	1.1218	90.6804	1.6019	54.8130	1.0000	0.8785	0.9544	0.9062	0.9629	1.0752	1.1770	0.9607	1.0641	1.3542	1.1698	0.9119	3.5431	1.0000	41.8809	2.5103	4.1862	0.8353	3.6829	69.2158
9	CAV	1.1228	1.0173	1.0965	64.0917	1.0000	68.3046	1.1299	0.7401	0.8999	0.8400	0.8317	1.1673	0.8125	0.8034	0.7003	0.9573	0.9933	0.9700	0.2923	1.0000	61.8240	3.7250	6.2129	1.2362	5.4642	8.9533
12	CAV	0.9331	1.0320	0.8555	48.3348	1.0000	98.5196	1.8992	0.7141	0.9270	1.2127	0.8574	0.8446	1.1886	1.1215	0.8833	1.0376	0.9837	0.8452	0.9239	1.0000	85.2781	5.1004	8.5090	1.7013	7.5180	36.9860
20	CAV	0.9307	0.9990	0.8863	33.4633	1.0000	70.2973	1.1741	0.8430	1.1583	1.0090	1.0746	0.8674	0.8812	1.0049	0.8714	0.9684	0.9353	1.1252	0.7149	1.0000	16.2218	0.9118	1.5200	0.3040	1.3423	4.3100
17	CAV	0.9085	1.3217	0.8341	41.2498	1.0000	93.3197	1.7580	1.1600	1.1378	0.9051	0.8815	1.1618	1.1501	0.9888	1.1124	0.8629	0.9051	0.9497	2.2988	1.0000	49.9461	2.9912	5.0070	0.9967	4.3980	49.3483
2	CAV	1.2633	1.0074	0.7388	90.1854	1.5005	77.1135	1.3390	1.2860	1.1166	1.2080	0.7610	0.9037	0.6272	0.7854	1.1652	0.9011	1.0021	0.7929	1.2092	1.0000	33.0545	1.9841	3.3102	0.6602	2.9145	16.0794
15	ECS	1.1383	1.0032	0.9306	90.4801	1.5972	46.0481	1.0000	1.1158	1.0209	1.0654	0.8872	1.0012	1.1768	0.9793	0.7447	0.9996	1.0030	0.9140	1.4788	1.0000	3.4630	0.2077	0.3466	0.0694	0.3049	1.6113
9	ECS	1.1115	0.8900	1.1800	89.0026	1.5047	34.3993	1.0000	0.9911	1.0043	0.9329	0.9197	1.0033	1.0119	0.7948	1.2583	0.9755	0.8679	1.1446	1.6263	1.0000	45.4001	2.7259	4.5370	0.9093	3.9941	75.3345
19	ECS	1.0905	0.7032	0.9816	94.5803	1.6924	51.0428	1.0000	1.2127	1.1632	1.2888	1.1231	1.1629	0.9548	1.3695	0.9939	0.9555	0.8499	1.2571	4.0080	1.0000	38.4068	2.3042	3.8405	0.7661	3.3786	601.8724
19	ECS	1.3972	0.9992	42.3906	1.0000	45.0606	1.0000	1.2380	1.0090	0.8169	0.9001	1.3177	1.0256	0.9440	0.9849	1.0731	0.8379	0.9049	1.3119	1.0000	7.3368	0.4383	0.7337	0.1464	0.6446	30.8128
9	ECS	1.0412	1.0730	1.0179	39.6553	1.0000	80.1378	1.4150	0.9879	1.1866	0.8625	0.8087	1.1148	1.0533	1.2097	0.9563	1.1708	0.9003	0.9987	1.8941	1.0000	72.3558	4.3453	7.2226	1.4472	6.3628	142.0752
5	ECS	0.8798	1.0611	0.9200	50.0012	1.0000	89.8633	1.6659	1.2871	0.8760	0.9412	1.0464	1.0900	1.0655	0.9863	1.1768	0.9709	1.0713	0.8102	1.7198	1.0000	98.9426	5.9359	9.8583	1.9824	8.6892	195.5091
10	ECS	0.9947	0.8069	1.1896	69.1097	1.1237	52.6697	1.0000	1.1602	1.0586	1.0995	0.9399	1.0887	1.3215	0.7388	0.8398	0.9715	0.9689	1.1296	1.2806	1.0000	47.8881	2.8698	4.7830	0.9574	4.2092	16.3853
10	ECS	1.1515	0.9691	1.0201	89.1720	1.5672	96.8333	1.8530	1.1380	1.1111	0.9021	0.9272	1.0848	0.9746	0.9745	0.9081	1.0699	0.9881	1.1986	4.1445	1.0000	25.2958	1.5168	2.5266	0.5070	2.2243	18.0413
11	ECS	0.7990	1.1883	0.9771	64.148:	1.0000	45.4744	1.0000	0.9990	0.9854	1.0452	0.8988	1.1000	1.1421	1.2423	1.2521	0.8507	1.1116	0.9262	1.4257	1.0000	32.5817	1.9560	3.2636	0.6531	2.8728	30.9809
5	ECS	1.2874	0.9569	0.9411	84.7167	1.4658	81.5567	1.4509	0.8165	0.9188	1.0774	1.1137	0.8312	1.0991	1.0749	1.3070	1.1000	0.8335	0.9988	1.0233	1.0000	19.0148	1.1379	1.8992	0.3807	1.6700	8.1477
6	ECS	1.1676	0.9703	1.1559	63.2752	1.0000	82.8878	1.4847	1.1569	0.7924	1.0824	0.6858	0.9167	1.1195	1.0095	0.7352	1.3116	0.9558	1.1429	0.8728	1.0000	27.5566	1.6482	2.7539	0.5492	2.4259	19.8350
4	ECS	1.0696	1.1574	1.1731	48.9400	1.0000	77.1192	1.3397	0.9863	1.0472	0.9991	0.7711	1.2291	0.9707	0.4470	1.0906	0.9074	0.9558	1.1321	0.7440	1.0000	24.7656	1.4903	2.4775	0.4939	2.1767	8.9225
15	ECS	0.8712	1.0273	0.8050	52.5458	1.0000	66.7460	1.0905	0.7757	1.0215	1.1136	0.9057	1.0029	0.8413	1.0082	0.9741	0.9577	1.3190	1.1321	0.7440	1.0000	38.6204	2.3202	3.8612	0.7681	3.3936	15.6834
4	ECS	0.6101	1.0723	1.1017	81.6465	1.3699	64.1070	1.0000	0.7667	1.2408	1.0093	0.9505	1.2193	0.9979	1.0028	1.1251	1.1192	0.6354	0.9377	0.8910	1.0000	92.4121	5.5282	9.2011	1.8457	8.1319	70.8552
2	ECS	1.1832	1.0213	0.9570	35.5145	1.0000	56.3848	1.0000	0.8269	1.0711	1.2155	0.9424	0.9149	1.0467	1.0785	1.0278	0.9132	1.0180	0.9950	0.8597	1.0000	54.7441	3.2795	5.4604	1.0916	4.8098	48.3076
12	ECS	1.0143	0.8431	1.1071	90.1410	1.5894	40.6059	1.0000	1.2553	1.3593	0.7902	0.8404	0.8582	1.1461	0.9072	1.0392	1.0098	0.9990	0.9557	1.5298	1.0000	89.5326	5.3644	8.9392	1.7859	7.8655	44.4221
17	ECS	0.9127	1.0617	0.8066	33.2360	1.0000	58.0402	1.0000	0.8214	1.0275	0.8834	0.7803	0.9805	1.1197	0.9457	1.0141	1.1177	0.9380	1.1547	0.5930	1.0000	6.0057	0.3614	0.6024	0.1204	0.5300	0.2828
18	ECS	0.8777	1.0204	1.0663	69.2282	1.1263	92.2018	1.7280	0.9701	1.1924	0.7942	0.8059	0.8804	0.8765	0.9138	0.9128	0.7453	1.1318	0.8708	0.6484	1.0000	8.7049	0.5219	0.8721	0.1745	0.7667	4.5844
15	ECS	1.0828	0.9343	1.0094	84.2650	1.4556	97.8481	1.8808	1.1577	0.7828	0.9030	0.8940	1.1220	0.8143	0.6199	1.0540	0.8703	1.1764	0.8692	1.0869	1.0000	47.4834	2.8547	4.7572	0.9492	4.1754	37.8238
1	ECS	0.9984	0.7427	1.0570	31.2127	1.0000	41.9807	1.0000	1.1236	0.9727	1.0450	1.0344	1.0302	0.9228	1.0571	0.8803	0.8424	1.0362	1.1569	0.8350	1.0000	47.5074	2.8547	4.7572	0.9492	4.1754	37.8238
14	OFF	0.7315	1.2034	0.9648	34.6562	1.0000	75.8445	1.3083	1.0001	0.7549	0.9800	1.1754	0.9744	0.9460	1.0375	0.9702	1.2558	1.1244	1.0741	1.2748	1.0000	67.5074	4.0903	6.7483	1.3472	5.9327	111.9861
2	OFF	1.0195	0.9841	0.9269	75.8992	1.2699	80.5285	1.4249	0.9021	0.9970	1.3136	0.9855	0.9949	0.9518	0.7749	1.0109	0.9517	1.0355	1.0894	1.5685	1.0000	59.2553	3.5598	5.9350	1.1891	5.2227	108.7757
19	OFF	0.8790	0.8190	0.8837	97.9443	1.7718	51.9652	1.0000	1.0763	0.8276	0.8459	1.1542	0.4536	1.2129	0.8991	0.5207	0.9488	0.9653	0.6628	0.1515	1.0000	8.5181	0.5151	0.8523	0.1705	0.7520	0.4978
9	OFF	1.2147	0.9705	1.0642	40.5925	1.0000	65.9311	1.0715	0.8002	0.5595	0.9253	1.1357	0.8710	1.1303	0.9079	0.8191	0.9389	1.1169	0.8873	0.4308	1.0000	46.3995	2.7920	4.6293	0.9234	4.0808	23.4449
20	OFF	1.2377	0.9521	0.9983	85.2439	1.4777	33.0463	1.0000	1.0403	0.9635	1.0083	0.6589	0.9901	1.0370	1.0144	1.1718	0.8100	1.1067	0.9726	1.2428	1.0000	99.3153	5.9696	9.9183	1.9788	8.7362	182.7377
13	OFF	0.9379	1.1205	1.1527	60.0972	1.0000	83.7746	1.5074	0.8068	0.9124	0.9803	0.8634	1.0838	0.8771	0.9945	0.7603	1.0709	0.8330	1.0051	0.6832	1.0000	38.9941	2.3324	3.8941	0.7828	3.4257	28.7338
11	OFF	1.0017	1.1167	0.9461	35.0497	1.0000	43.4237	1.0000	1.1520	0.9887	1.1682	0.8844	1.0283	0.7242	0.9392	0.9865	1.0970	1.1216	1.2178	1.2883	1.0000	32.4769	1.9584	3.2469	0.6487	2.8626	44.9711
11	OFF	0.8883	1.0993	1.2669	76.7179	1.2878	97.1448	1.8615	0.7310	1.0170	1.0332	1.1675	1.1030	1.0842	0.9475	1.0183	0.9722	1.0298	1.2218	2.6727	1.0000	92.1173	5.5354	9.2010	1.8509	8.1075	341.0478
11	OFF	0.9053	1.0382	1.2482	73.4305	1.2194	54.9332	1.0000	1.1112	1.0341	1.4163	1.1925	0.8666	0.8764	1.2056	0.9352	0.9844	1.0285	0.9467	2.2687	1.0000	25.5812	1.5309	2.5552	0.5100	2.2472	87.5597
4	OFF	0.9623	0.9799	1.1673	50.2594	1.0000	86.6277	1.5811	1.1931	0.8118	1.0097	0.6900	1.1483	1.0289	1.2432	0.9763	0.8491	1.2159	0.7163	1.2455	1.0000	85.8471	5.1825	8.5936	1.7135	7.5750	199.0331
5	OFF	1.3290	1.1458	0.8340	63.9590	1.0000	56.9109	1.0000	0.8835	1.0406	0.7582	0.7334	1.2050	1.2632	0.9103	0.8277	1.6902	1.0243	1.3094	1.6878	1.0000	21.2062	1.2735	2.1187	0.4224	1.8653	41.4659
1	OFF	1.1016	0.9674	1.1708	98.0244	1.7733	50.3092	1.0000	1.0479	0.9373	1.1708	1.0338	1.1595	1.1787	1.1093	0.8049	1.1720	1.3026	0.7455	3.7659	1.0000	25.2618	1.5141	2.5167	0.5053	2.2270	82.7360
3	OFF	0.9824	0.8182	1.1304	97.6717	1.7649	56.1926	1.0000	0.9601	1.0158	1.0300	0.9523	0.8171	1.1576	0.8168	1.1217	1.1495	1.1764	0.9415	1.6924	1.0000	86.9301	5.1970	8.7203	1.7395	7.6540	201.7803
14	OFF	0.9484	0.9195	1.0932	74.8235	1.2465	73.7752	1.2577	1.2050	1.0043	1.0946	1.1144	1.0506	1.2087	1.0344	0.8142	0.9614	0.8827	0.8697	1.8192	1.0000	40.4395	2.4110	4.0404	0.8069	3.5488	89.6710
19	OFF	1.0762	0.9047	0.8553	73.1294	1.2098	77.1144	1.3396	0.9181	1.1020	0.8759	1.0488	0.9970	1.3295	1.1530	1.1224	0.9486	1.1870	1.0109	1.8891	1.0000	86.4582	5.1892	8.6928	1.7262	7.6171	80.5726
18	OFF	1.0142	1.1617	0.9627	62.9555	1.0000	50.1117	1.0000	0.9816	0.8612	1.0422	0.9824	0.8534	1.0824	0.8045	1.2343	0.7865	0.8927	0.6358	0.4470	1.0000	86.4582	5.1892	8.6928	1.7262	7.6171	80.5726
12	OFF	1.4246	1.1480	1.0057	46.3868	1.0000	91.5121	1.7097	0.7808	1.2163	1.1070	0.8528	0.9781	1.2003	1.2041	1.1744	1.2241	0.8811	1.0020	4.5265	1.0000	23.9870	1.4419	2.3878	0.4779	2.1062	73.7434
15	OFF	1.0894	0.8458	0.9071	92.2270	1.6376	99.9046	1.9374	0.9492	1.2075	0.9904	0.9902	1.2519	0.9924	0.9900	1.1916	0.8925	0.8627	0.8901	2.2263	1.0000	43.9940	2.6006	4.3304	0.8681	3.8173	110.6524
6	OFF	1.0731	1.1847	1.0824	90.9986	1.6092	88.7218	1.6359	0.9238	0.9408	1.0021	1.0774	0.9309	0.9797	1.0878	0.9487	0.7442	0.9334	0.9289	2.1258	1.0000	55.4763	3.3223	5.5475	1.1059	4.8776	138.5903
14	OFF	0.8991	0.7009	0.7305	69.9427	1.1415	81.2399	1.4429	1.2130	0.8993	1.0000	1.0527	0.8560	0.9340	0.9096	0.9032	0.7881	0.9436	1.0726	0.4994	1.0000	16.3312	0.9815	1.6257	0.3288	1.4348	6.1412

Composite Database

Navigation & Core Avionics Categories

BCNUM	CATEGORY	RELY	DBSIZE	CPLX	TIMEUTIL	EXTIME	MEMUTIL	STOR	VIRT	TURN	ACAP	AEXP	PCAP	VEXP	LEXP	MODP	TOOL	SCED	RVOL	PI	ENTROPY	ACTIDSI	ADDIDSI	MODIDSI	DELIDSI	EDSI	ACTEFFORT
3	NAV	1.1922	0.6108	0.9400	30.9388	1.0000	62.5242	1.0000	1.0068	1.0228	1.0350	0.9423	1.2346	0.7468	0.7764	1.3720	1.1629	0.8192	1.0342	0.6652	1.0000	0.3498	0.0210	0.0349	0.0070	0.0307	0.2074
17	NAV	1.0828	0.9782	1.0460	31.1534	1.0000	37.1985	1.0000	1.2159	1.2964	1.2133	1.1666	0.7904	0.9047	0.9609	1.0769	1.0317	0.9221	0.9959	1.9209	1.0000	91.8841	5.8207	9.2440	1.8381	8.1183	92.4483
1	NAV	1.3095	1.0678	0.9977	73.0298	1.2076	76.0460	1.3132	1.0421	0.9930	1.0816	1.1148	0.6974	0.9803	0.9085	0.9851	1.0803	0.9221	0.8626	1.0454	1.0000	96.5974	5.8120	9.6025	1.9344	8.4805	49.1518
11	NAV	0.8783	1.0002	0.7905	63.1843	1.4313	84.3082	1.5211	0.7954	0.9773	1.2031	1.0128	1.1167	1.0185	0.7163	1.1060	0.9779	1.3431	0.7700	1.7736	1.0000	26.8432	1.6086	2.6821	0.5360	2.3568	19.9706
6	NAV	0.9599	0.8324	0.9853	99.9517	1.5851	81.9601	1.4609	0.9800	0.9654	0.8397	1.2815	1.2979	0.6517	0.9107	0.9888	0.9411	1.0323	1.2102	1.6627	1.0000	90.2876	5.4092	9.0329	1.7979	7.9411	75.6926
18	NAV	0.8131	0.8176	1.1357	48.3859	1.0000	98.8106	1.9072	1.0463	1.2849	1.1711	1.2543	0.8360	1.0666	1.2596	0.8474	1.1932	1.0477	1.2915	4.4511	1.0000	39.9090	2.3910	4.0036	0.7991	3.6143	78.1694
2	NAV	0.9746	1.0008	0.5484	85.2593	1.4781	69.5723	1.1568	1.0381	1.1316	1.0434	1.1484	1.2347	0.9481	0.9725	0.8111	1.0723	0.9264	1.0015	1.5049	1.0000	23.9985	1.4439	2.4014	0.4810	2.1146	11.7727
6	NAV	0.6451	1.0224	1.0270	44.0331	1.0000	78.7616	1.3806	1.0350	1.3064	0.9911	0.8707	1.0402	0.9072	0.9364	1.2931	1.3023	0.9264	1.0015	1.5049	1.0000	36.1697	2.1700	3.6136	0.7238	3.1817	25.0040
20	NAV	1.0657	1.0322	1.0645	80.0072	1.3404	45.7486	1.0000	0.8921	1.2234	1.2588	1.0458	1.0283	1.2323	1.0789	1.1315	1.0771	1.3225	1.0957	5.4755	1.0000	65.3084	3.9384	6.5598	1.3008	5.7557	167.7160
6	NAV	1.0151	1.1151	0.6928	65.0204	1.0378	39.7472	1.0000	0.9996	1.0022	0.8829	0.9124	1.1075	1.2217	0.5388	1.0563	0.8871	0.9660	0.9265	0.9714	1.0000	45.6732	2.7384	4.5603	0.9139	4.0147	9.9973
12	NAV	1.2003	0.9904	1.0775	42.7501	1.0000	34.8149	1.0000	0.8851	0.8156	1.0450	1.0930	1.1650	0.8672	1.0465	0.8287	0.9097	0.7313	1.0953	0.6267	1.0000	6.7952	0.4087	0.6810	0.1357	0.5995	1.4788
16	NAV	1.1043	0.8630	0.9667	53.5625	1.0000	84.8532	1.5352	1.1094	1.0818	1.2714	1.0173	0.9998	1.2955	1.1285	0.8096	0.9483	1.2962	1.1991	2.6788	1.0000	23.5279	1.4033	2.3666	0.4716	2.0582	24.3480
20	NAV	0.8704	1.1099	0.9848	99.5426	1.8091	78.6072	1.3767	0.8271	0.9015	0.8203	0.8427	1.0431	0.9340	0.7620	0.6409	0.9010	0.9927	1.0109	0.3854	1.0000	66.0372	3.9843	6.5799	1.3168	5.8093	12.9799
18	NAV	1.2156	0.9559	1.1004	39.4941	1.0000	54.6657	1.0000	0.9346	1.2249	1.0730	0.9760	1.0991	1.3643	0.8227	1.0225	0.9884	0.9777	0.8239	1.5080	1.0000	49.8017	2.9961	4.9700	0.9964	4.3917	38.7521
7	NAV	1.2745	1.2635	1.1977	43.9906	1.0000	90.4613	1.8429	0.9857	0.8560	1.4490	1.0637	1.0412	0.9468	1.0865	0.8249	1.2707	0.9317	1.0216	4.7837	1.0000	85.6124	5.1261	8.5534	1.7056	7.5271	183.1132
17	NAV	0.7323	0.9213	0.6077	70.1200	1.1452	87.2705	1.5979	0.9126	1.1707	1.1097	1.0129	1.0834	1.0954	0.9373	1.0594	1.0195	1.1901	1.1155	1.3643	1.0000	29.5728	1.7670	2.9498	0.5917	2.6050	19.5912
2	NAV	0.8677	0.8040	0.6499	36.9955	1.0000	95.2133	1.8090	0.9481	1.2502	1.0274	0.9974	1.2652	1.0870	0.8228	1.3163	1.0393	1.0782	0.8526	1.5009	1.0000	33.0990	1.9797	3.3066	0.6614	2.9076	17.3372
7	NAV	0.9061	1.1012	0.9427	70.2448	1.1479	58.2536	1.0000	0.7840	1.0408	1.0824	1.1506	0.9271	1.1373	0.7941	1.1599	1.0378	0.8355	0.7434	0.6929	1.0000	92.9194	5.5556	9.2553	1.8456	8.1499	33.3095
1	NAV	0.7882	0.9400	0.8078	47.2652	1.0000	99.1990	1.9179	1.2658	0.9159	1.0900	0.9786	0.9430	0.8570	0.9172	0.8503	0.9732	0.9407	0.8239	0.7159	1.0000	1.0751	0.0644	0.1073	0.0216	0.0945	0.0174
5	NAV	0.9364	0.9733	0.9168	52.1731	1.0000	70.7009	1.1837	0.9690	1.1974	1.2292	1.0356	1.2455	1.1850	1.3829	0.9283	1.1749	1.0847	0.7962	2.8122	1.0000	94.9195	5.6887	9.4317	1.8925	8.3172	128.1715
2	CAV	0.9628	1.2726	0.9046	60.8034	1.0000	80.2545	1.4180	1.2294	1.1161	0.9152	0.9866	0.8405	0.8443	0.9853	1.2949	0.9830	1.1637	1.0032	2.0713	1.0000	7.1188	0.4267	0.7152	0.1419	0.6278	1.6507
11	CAV	0.7916	1.0804	0.9193	45.7098	1.0000	35.1985	1.0000	0.8320	1.0363	1.0867	0.8319	0.8777	1.0340	1.2183	0.9129	0.6224	1.0699	1.2391	0.5086	1.0000	15.6514	0.9407	1.5653	0.3121	1.3780	2.7540
11	CAV	0.8258	0.7369	1.0156	81.1647	1.3861	97.7155	1.9772	0.9287	1.1267	0.8094	0.9417	0.8486	1.1476	1.0216	0.9419	1.2990	0.8303	0.9264	0.8382	1.0000	93.8063	5.6211	9.4130	1.8776	8.2979	38.8087
14	CAV	1.2328	1.2043	0.9414	98.1594	1.7764	89.8321	1.6651	1.2780	1.1049	1.2537	1.0922	1.2048	1.2458	1.4120	0.8497	0.9993	0.8768	0.6569	0.6674	1.0000	60.1930	3.5986	6.0078	1.2061	5.2794	209.6187
16	CAV	0.9661	0.9459	0.9244	59.4038	1.0000	88.4182	1.6279	1.0570	1.2735	0.7450	1.6307	1.2354	0.8882	1.2012	0.8073	0.9043	0.8844	1.1816	2.2629	1.0000	28.7216	1.7242	2.8772	0.5770	2.5312	25.7600
5	CAV	1.2065	1.1328	1.0997	64.3266	1.0000	40.1878	1.0000	1.1172	1.1261	1.0707	0.9428	1.1486	1.2053	1.2134	0.9784	0.7604	1.1446	0.8926	2.3036	1.0000	58.6226	3.5110	5.8579	1.1711	5.1528	68.6709
12	CAV	1.0355	1.1042	0.9761	34.0754	1.0000	76.6984	1.3293	1.0484	0.7593	1.2337	0.8904	1.1459	1.0158	0.7756	0.7978	0.6413	1.1923	0.8361	0.8556	1.0000	4.4370	0.2670	0.4451	0.0882	0.3904	2.1038
1	CAV	1.1751	1.1584	0.9655	96.5102	1.7377	59.3514	1.0000	1.0682	1.4239	1.1360	0.9125	0.8802	0.8256	1.2856	0.7469	0.8684	1.3293	0.9747	2.9381	1.0000	66.0182	3.9876	6.6019	1.3137	5.8219	92.6930
18	CAV	1.1995	1.0542	0.8073	61.2341	1.0000	34.3920	1.0000	1.0365	0.8351	1.2601	1.1480	1.2848	0.9182	1.1262	0.8817	1.0016	1.0333	0.8915	1.3816	1.0000	38.5324	2.3095	3.8529	0.7757	3.3906	20.6944
19	CAV	0.9500	1.2294	1.3165	84.5328	1.4616	57.9090	1.0000	0.9328	0.9878	0.8770	1.1164	1.1785	0.9726	0.7222	0.7391	0.9705	0.9140	0.8662	0.9677	1.0000	62.4581	3.7668	6.2316	1.2504	5.4997	27.4553
19	CAV	0.9174	0.7194	0.8299	74.4348	1.2381	90.8350	1.6917	0.9612	0.6712	0.9109	0.7807	0.8116	0.8853	0.9839	1.0261	1.1558	0.9824	1.1733	0.5015	1.0000	73.5862	4.3904	7.3085	1.4799	6.4690	16.6944
20	CAV	1.0460	0.9779	0.8370	78.6812	1.2682	33.9464	1.0000	1.0110	0.9682	0.9864	0.7811	0.9569	0.9101	0.9872	0.8786	1.0719	0.9609	1.0781	0.6745	1.0000	53.0602	3.1815	5.3716	1.0880	4.6998	15.3174
3	CAV	1.2024	1.0577	0.8408	94.7320	1.6989	42.8534	1.0000	1.3376	1.0837	1.1859	1.1987	1.0000	1.2124	1.0010	1.2849	1.1984	0.5331	0.8124	3.0240	1.0000	77.1763	4.6275	7.7129	1.5448	6.7878	101.2718
9	CAV	1.1839	0.9881	0.9018	64.7689	1.0000	73.0497	1.2401	1.0900	0.7303	0.9995	0.7981	1.1274	0.9205	0.9353	0.7890	0.9994	1.0686	0.7873	0.6009	1.0000	67.7828	4.0725	6.7990	1.3532	5.9524	15.4225
9	CAV	1.4488	0.9969	1.1218	90.6804	1.6019	54.8130	1.0000	0.8785	0.9544	0.9062	0.9829	1.0782	1.1770	0.9967	1.0041	1.3542	1.1608	0.9119	3.5431	1.0000	41.8809	2.5103	4.1852	0.8353	3.6829	69.2158
12	CAV	0.9331	1.0320	0.8555	48.3348	1.0000	98.5196	1.8692	0.7141	0.9270	1.2127	0.8574	0.8446	1.1886	1.1215	0.8833	1.0376	0.9837	0.8452	0.9239	1.0000	85.2781	5.1094	8.5690	1.7013	7.5189	36.9860
20	CAV	0.9307	0.9990	0.8963	33.4633	1.0000	70.2973	1.1741	0.8430	1.1583	1.0000	1.0746	0.8874	0.8812	1.0049	0.8714	0.9684	0.9353	1.1252	0.7149	1.0000	15.2218	0.9118	1.5290	0.3049	1.3423	4.3100
17	CAV	0.9985	1.3217	0.8341	41.2498	1.0000	93.3197	1.7580	1.1600	1.1378	0.9061	0.8815	1.1618	1.1501	0.9988	1.1124	0.8929	0.9061	0.9497	2.2988	1.0000	49.9461	2.9912	5.0070	0.9967	4.3980	49.3483
2	CAV	1.2633	1.0974	0.7368	90.1854	1.5405	77.1135	1.3596	1.2860	1.1166	1.2080	0.7610	0.9637	0.8272	0.7854	1.1682	0.9011	1.0821	0.7929	1.2092	1.0000	33.0545	1.9841	3.3102	0.6602	2.9145	16.0794

Series Database

5% Growth

BCNUM	CATEGORY	RELY	DBSIZE	CPLX	TIMEUTL	MEMUTL	EXTIME	STOR	VIRT	TURN	ACAP	AEXP	PCAP	VEXP	LEXP	MODP	TOOL	SCED	RVOL	PI	ENTROPY	ACTDSI	ADDKDSI	MODKDSI	DELKDSI	EDSI	ACTEFFORT
1	CAV	0.8484	1.1109	1.1462	50.0000	50.0000	1.0000	1.0000	0.9772	1.1744	1.4884	1.1086	1.2109	1.1599	0.9334	0.8334	1.1128	0.6270	1.1532	1.7964	1.0000	100.0000	4.9995	10.0015	1.9942	8.2690	105.2635
2	CAV	0.8484	1.1109	1.1462	51.5026	51.5026	1.0000	1.0000	0.9772	1.1744	1.4884	0.9735	1.2109	1.1599	0.8197	0.8334	1.1128	0.6270	1.1532	1.3852	1.0000	103.0052	5.1772	10.2914	2.0719	8.5311	77.8898
3	CAV	0.8484	1.1109	1.1462	53.0557	53.0557	1.0000	1.0000	0.9772	1.1744	1.4884	0.9222	1.2109	1.1599	0.7997	0.8334	1.1128	0.6270	1.1532	1.1998	1.0000	106.1114	5.3293	10.5799	2.1320	8.7694	64.1791
4	CAV	0.8484	1.1109	1.1462	54.6503	54.6503	1.0000	1.0000	0.9772	1.1744	1.4884	0.8546	1.2109	1.1599	0.7196	0.8334	1.1128	0.6270	1.1532	1.0681	1.0000	109.3207	5.4929	10.9265	2.1787	9.0343	69.0478
5	CAV	0.8484	1.1109	1.1462	56.3025	56.3025	1.0000	1.0000	0.9772	1.1744	1.4884	0.8198	1.2109	1.1599	0.6603	0.8334	1.1128	0.6270	1.1532	0.9824	1.0000	112.6049	5.6097	11.2619	2.2460	9.3004	67.7203
6	CAV	0.8484	1.1109	1.1462	57.9948	57.9948	1.0000	1.0000	0.9772	1.1744	1.4884	0.7923	1.2109	1.1599	0.6071	0.8334	1.1128	0.6270	1.1532	0.9175	1.0000	115.9695	5.8097	11.5977	2.3823	9.5725	61.8024
7	CAV	0.8484	1.1109	1.1462	59.7385	59.7385	1.0000	1.0000	0.9772	1.1744	1.4884	0.7923	1.2109	1.1599	0.6071	0.8334	1.1128	0.6270	1.1532	0.9175	1.0000	119.4770	5.9942	11.9429	2.4003	9.8716	61.6474
8	CAV	0.8484	1.1109	1.1462	61.5556	61.5556	1.0000	1.0000	0.9772	1.1744	1.4884	0.7923	1.2109	1.1599	0.6071	0.8334	1.1128	0.6270	1.1532	0.9175	1.0000	123.0712	6.1955	12.2869	2.4545	10.1881	73.4792
9	CAV	0.8484	1.1109	1.1462	63.4061	63.4061	1.0000	1.0000	0.9772	1.1744	1.4884	0.7923	1.2109	1.1599	0.6071	0.8334	1.1128	0.6270	1.1532	0.9175	1.0000	126.8123	6.3459	12.6726	2.5414	10.4876	99.2244
10	CAV	0.8484	1.1109	1.1462	65.3083	65.3083	1.0439	1.0671	0.9772	1.1744	1.4884	0.7923	1.2109	1.1599	0.6071	0.8334	1.1128	0.6270	1.1532	1.0123	1.0000	130.6167	6.5440	12.9967	2.6197	10.7874	69.9927
11	CAV	0.8484	1.1109	1.1462	67.2805	67.2805	1.0861	1.1029	0.9772	1.1744	1.4884	0.7923	1.2109	1.1599	0.6071	0.8334	1.1128	0.6270	1.1532	1.0980	1.0000	134.5610	6.7245	13.4495	2.6996	11.1228	86.9855
12	CAV	0.8484	1.1109	1.1462	69.2934	69.2934	1.1277	1.1502	0.9772	1.1744	1.4884	0.7923	1.2109	1.1599	0.6071	0.8334	1.1128	0.6270	1.1532	1.1900	1.0000	138.5899	6.9709	13.8154	2.7956	11.4213	122.0723
13	CAV	0.8484	1.1109	1.1462	71.3671	71.3671	1.1717	1.1994	0.9772	1.1744	1.4884	0.7923	1.2109	1.1599	0.6071	0.8334	1.1128	0.6270	1.1532	1.2993	1.0000	142.7141	7.1668	14.1963	2.8473	11.7718	103.9427
14	CAV	0.8484	1.1109	1.1462	73.5198	73.5198	1.2182	1.2514	0.9772	1.1744	1.4884	0.7923	1.2109	1.1599	0.6071	0.8334	1.1128	0.6270	1.1532	1.3986	1.0000	147.0336	7.3389	14.4847	2.9472	12.1462	124.2740
15	CAV	0.8484	1.1109	1.1462	75.7129	75.7129	1.2599	1.3050	0.9772	1.1744	1.4884	0.7923	1.2109	1.1599	0.6071	0.8334	1.1128	0.6270	1.1532	1.5156	1.0000	151.4258	7.5800	14.7959	3.0812	12.5396	129.5156
16	CAV	0.8484	1.1109	1.1462	77.9722	77.9722	1.3154	1.3607	0.9772	1.1744	1.4884	0.7923	1.2109	1.1599	0.6071	0.8334	1.1128	0.6270	1.1532	1.6423	1.0000	155.9444	7.8266	15.4080	3.1218	12.9192	148.1012
17	CAV	0.8484	1.1109	1.1462	80.3240	80.3240	1.3674	1.4197	0.9772	1.1744	1.4884	0.7923	1.2109	1.1599	0.6071	0.8334	1.1128	0.6270	1.1532	1.7811	1.0000	160.6481	8.0733	16.0812	3.2083	13.2934	210.3978
18	CAV	0.8484	1.1109	1.1462	82.7316	82.7316	1.4211	1.4808	0.9772	1.1744	1.4884	0.7923	1.2109	1.1599	0.6071	0.8334	1.1128	0.6270	1.1532	1.9300	1.0000	165.4631	8.2729	16.4900	3.2927	13.6614	285.6405
19	CAV	0.8484	1.1109	1.1462	85.2315	85.2315	1.4774	1.5449	0.9772	1.1744	1.4884	0.7923	1.2109	1.1599	0.6071	0.8334	1.1128	0.6270	1.1532	2.0941	1.0000	170.4630	8.5611	17.0423	3.4035	14.1086	266.5045
20	CAV	0.8484	1.1109	1.1462	87.8053	87.8053	1.5369	1.6118	0.9772	1.1744	1.4884	0.7923	1.2109	1.1599	0.6071	0.8334	1.1128	0.6270	1.1532	2.2713	1.0000	175.6105	8.8033	17.5364	3.5057	14.5508	284.9212

Series Database

6% Growth

SCNUM	CATEGORY	RELY	DBSIZE	CPLX	TIMEUTIL	EXTIME	MEMUTIL	STOR	VIRT	TURN	ACAP	AEXP	PCAP	VEXP	LEXP	MODP	TOOL	SCED	RVOL	PI	ENTROPY	ACTIDSI	ADDIDSI	MODIDSI	DELIDSI	EDSI	ACTEFORT
1	CAV	1.0772	1.2808	0.8400	50.0000	1.0000	50.0000	1.0000	1.4796	1.1021	0.7426	1.2181	0.9991	1.2988	0.9049	0.7629	0.8667	0.7795	1.2097	1.1999	1.0000	100.0000	6.0154	9.9922	2.0113	8.8076	76.9946
2	CAV	1.0772	1.2808	0.8400	52.0025	1.0000	52.0025	1.0000	1.4796	1.1021	0.7426	1.0977	0.9991	1.2988	0.7946	0.7629	0.8667	0.7795	1.2097	0.9253	1.0000	104.0051	6.2426	10.4081	2.0928	9.1682	83.5498
3	CAV	1.0772	1.2808	0.8400	54.0823	1.0000	54.0823	1.0000	1.4796	1.1021	0.7426	0.9914	0.9991	1.2988	0.7344	0.7629	0.8667	0.7795	1.2097	0.7948	1.0000	108.1647	6.4737	10.8086	2.1824	9.5000	88.0336
4	CAV	1.0772	1.2808	0.8400	56.2430	1.0000	56.2430	1.0000	1.4796	1.1021	0.7426	0.8993	0.9991	1.2988	0.6778	0.7629	0.8667	0.7795	1.2097	0.7135	1.0000	112.4860	6.7448	11.2391	2.2478	9.8722	94.3515
5	CAV	1.0772	1.2808	0.8400	58.4725	1.0000	58.4725	1.0000	1.4796	1.1021	0.7426	0.9008	0.9991	1.2988	0.6492	0.7629	0.8667	0.7795	1.2097	0.6882	1.0000	116.9850	7.0288	11.7294	2.3372	10.3133	99.7002
6	CAV	1.0772	1.2808	0.8400	60.8494	1.0000	60.8494	1.0000	1.4796	1.1021	0.7426	0.8705	0.9991	1.2988	0.6467	0.7629	0.8667	0.7795	1.2097	0.6128	1.0000	121.6908	7.2776	12.1926	2.4307	10.6799	47.1154
7	CAV	1.0772	1.2808	0.8400	63.2856	1.0000	63.2856	1.0000	1.4796	1.1021	0.7426	0.8705	0.9991	1.2988	0.6467	0.7629	0.8667	0.7795	1.2097	0.6128	1.0000	126.5192	7.5699	12.6803	2.5240	11.1167	49.9240
8	CAV	1.0772	1.2808	0.8400	65.8023	1.0541	65.8023	1.0946	1.4796	1.1021	0.7426	0.8705	0.9991	1.2988	0.6467	0.7629	0.8667	0.7795	1.2097	0.6403	1.0000	131.6045	7.8942	13.1568	2.6462	11.5614	47.2594
9	CAV	1.0772	1.2808	0.8400	68.4263	1.1093	68.4263	1.1296	1.4796	1.1021	0.7426	0.8705	0.9991	1.2988	0.6467	0.7629	0.8667	0.7795	1.2097	0.7680	1.0000	136.8526	8.2121	13.6917	2.7362	12.0459	85.5065
10	CAV	1.0772	1.2808	0.8400	71.1642	1.1676	71.1642	1.1947	1.4796	1.1021	0.7426	0.8705	0.9991	1.2988	0.6467	0.7629	0.8667	0.7795	1.2097	0.8649	1.0000	142.3284	8.5398	14.2123	2.8456	12.5477	81.6701
11	CAV	1.0772	1.2808	0.8400	74.0110	1.2299	74.0110	1.2634	1.4796	1.1021	0.7426	0.8705	0.9991	1.2988	0.6467	0.7629	0.8667	0.7795	1.2097	0.9615	1.0000	148.0221	8.8809	14.7802	2.9585	13.0296	83.0079
12	CAV	1.0772	1.2808	0.8400	76.9077	1.2793	76.9077	1.3350	1.4796	1.1021	0.7426	0.8705	0.9991	1.2988	0.6467	0.7629	0.8667	0.7795	1.2097	1.0669	1.0000	153.9353	9.2540	15.3805	3.0766	13.5486	114.4044
13	CAV	1.0772	1.2808	0.8400	80.0564	1.3416	80.0564	1.4130	1.4796	1.1021	0.7426	0.8705	0.9991	1.2988	0.6467	0.7629	0.8667	0.7795	1.2097	1.1700	1.0000	160.1128	9.6906	15.9906	3.2073	14.0769	192.6122
14	CAV	1.0772	1.2808	0.8400	83.2628	1.4328	83.2628	1.4941	1.4796	1.1021	0.7426	0.8705	0.9991	1.2988	0.6467	0.7629	0.8667	0.7795	1.2097	1.3119	1.0000	166.5060	10.0089	16.6094	3.3227	14.6373	160.9436
15	CAV	1.0772	1.2808	0.8400	86.5941	1.5083	86.5941	1.5802	1.4796	1.1021	0.7426	0.8705	0.9991	1.2988	0.6467	0.7629	0.8667	0.7795	1.2097	1.4607	1.0000	173.1882	10.3967	17.2218	3.4528	15.1928	188.8328
16	CAV	1.0772	1.2808	0.8400	90.0560	1.5876	90.0560	1.6710	1.4796	1.1021	0.7426	0.8705	0.9991	1.2988	0.6467	0.7629	0.8667	0.7795	1.2097	1.6267	1.0000	180.1121	10.8007	17.8906	3.6043	15.8443	190.1967
17	CAV	1.0772	1.2808	0.8400	93.6587	1.6709	93.6587	1.7671	1.4796	1.1021	0.7426	0.8705	0.9991	1.2988	0.6467	0.7629	0.8667	0.7795	1.2097	1.8096	1.0000	187.3176	11.2080	18.7362	3.7799	16.4619	284.8385
18	CAV	1.0772	1.2808	0.8400	97.3914	1.7583	97.3914	1.8683	1.4796	1.1021	0.7426	0.8705	0.9991	1.2988	0.6467	0.7629	0.8667	0.7795	1.2097	2.0132	1.0000	194.7827	11.7172	19.4884	3.8897	17.1606	297.4593
19	CAV	1.0772	1.2808	0.8400	98.6957	1.7991	98.6957	1.9040	1.4796	1.1021	0.7426	0.8705	0.9991	1.2988	0.6467	0.7629	0.8667	0.7795	1.2097	2.0877	1.0000	202.6102	12.2098	20.3497	4.0549	17.9056	286.4337
20	CAV	1.0772	1.2808	0.8400	99.3478	1.8045	99.3478	1.9220	1.4796	1.1021	0.7426	0.8705	0.9991	1.2988	0.6467	0.7629	0.8667	0.7795	1.2097	2.1255	1.0000	210.7542	12.6389	21.0624	4.2061	18.5236	301.2316

Bibliography

1. Babel, Philip S. Class handout, Colloquium, Systems Software Management. School of Systems and Logistics, Air Force Institute of Technology, Wright-Patterson AFB OH, 11 May 1993.

2. Banker, Rajiv D., Srikant M Datar, and Chris F. Kemerer. "A Model to Evaluate Variables Impacting the Productivity of Software Maintenance Projects," *Management Science*, 37: 1 - 18 (1 January 1991).

3. Barber, Brent L. *Investigative Search of Quality Historical Software Support Cost Data and Software Support Cost-Related Data.* MS thesis, AFIT/GSS/LSY/91D-1. School of Systems and Logistics, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, December 1991 (AD-A246659).

4. Barrow, Dean, Susan Nilson, and Dawn Timberlake. *Software Estimation Technology Report*, Hill AFB UT: Software Technology Center (STSC), March 1993.

5. Boehm, Barry W. *Software Engineering Economics*, Englewood Cliffs NJ: Prentice-Hall, Inc., 1981.

6. Boulware, Gary W., Belinda J. Nethery, and Bryan D. Turner. "Maintenance Software Model Assumptions Versus Budgeting Realities," *National Estimator*, Spring: 13 - 25 (1991).

7. Bourque, Pierre and Vianney Côté. "An Experiment in Software Sizing with Structured Analysis Metrics," *Journal of Systems and Software*, 15 No2: 159 - 172 (May 1991).

8. Brooks, Frederick P., Jr. *The Mythical Man Month*, Reading MA: Addison-Wesley, 1982.

9. Cioch, Frank A. "Measuring Software Misinterpretation," *Journal of Systems Software*, 14: 85 - 95 (February 1991).

10. Compton, B. Terry and Carol Withrow. "Prediction and Control of Ada Software Defects," *Journal of Systems and Software*, 12 No3: 199 - 207 (July 1990).

11. Corbi, T. A. "Program Understanding: Challenge for the 1990s," *IBM Systems Journal*, 28 No2: 294 - 305 (1989).

12. Defense Systems Management College. *Mission Critical Computer Resources Management Guide*. Washington: GPO, 22 May 1990 (AD-A264652).

13. Department of Defense. *Defense System Software Development*. DOD-STD-2167A. Washington: GPO, 29 February 1988.

14. Department of Defense. *Mission-Critical Computer Resources Software Support*. MIL-HDBK-347. Washington: GPO, 22 May 1990.

15. Devore, Jay L. *Probability and Statistics for Engineering and the Sciences*. Pacific Grove CA: Brooks/Cole Publishing Company, 1991.

16. Eddins-Earles, Mary. *C^3I Software Cost Estimation Model Development, Final Technical Report, October 1984 - May 1987*. Contract F360602-84-C-0154. Griffis AFB: Rome Air Development Center, September 1987 (AD-B120 201).

17. Enhanced REVIC Advisor (ENREV). Version 2.01 , IBM, 258k, disk. Computer Software and manual. Keith Ernst, 1991.

18. Fenton, N. E. *Software Metrics: A Rigorous Approach*. London: Chapman & Hall, 1991.

19. Ferens, Daniel V. "Evaluation of Eight Software Support Cost Models," *National Estimator*, Spring: 3 - 12 (1991).

20. Ferens, Daniel V. "New Perspectives in Software Logistics Support," *Logistics Spectrum*, 26: 4 - 8 (Spring 1992).

21. Ferens, Daniel V. "Software Cost Models: Quo Vadis," *Journal of Parametrics*, 4 No4: 64 - 79 (December 1984).

22. Fried, Louis. "Team Size and Productivity in Systems Development," *Journal of Information Systems Management*, 8 No3: 27 - 35 (1991).

23. Glass, Robert L. and Ronald A Noiseux. *Software Maintenance Guidebook*. Englewood Cliffs NJ: Prentice-Hall, 1981

24. Greve, Alan R. and others. *The REVIC Advisor (REVAD): An Expert System Preprocessor to a Parametric Software Cost Estimating Model.* Alexandria VA: DOD Defense Logistics Agency, September 1991 (AD-A242707).

25. Gulezian, Ronald. "Reformulating and Calibrating COCOMO," *Journal of Systems and Software*, 16 No3: 235 - 242 (November 1991).

26. Hager, James A. "Software Cost Reduction Methods in Practice," *IEEE Transactions on Software Engineering*, 15 No12: 1638 - 1644 (December 1989).

27. Henry, Sallie and Calvin Selig. "Predicting Source-Code Complexity at the Design Stage," *IEEE Software*, 7: 36 - 44 (March 1990).

28. Kane, Patrick T., Donald J. Reifer, and Douglas Willens. *SoftCost-R, Software Version 8.0, Manual Revision - October 1989.* Torrance CA: Reifer Consultants. Inc., 1989.

29. Kankey, Roland D. "An Overview of Software Maintenance Costing, *National Estimator*, Spring: 40 - 47 (1991).

30. Lederer, Albert L. and Jayesh Prasad. "Nine Management Guidelines for Better Cost Estimating," *Communications of the ACM*, 35 No2: 51 - 59 (February 1992).

31. Lederer, Albert L. and Jayesh Prasad. "The Validation of a Political Model of Information Systems Development Cost Estimating," *Computer Personnel*, 13 No2: 47 - 57 (July 1991).

32. Lehner, Franz. "Cost Comparison for the Development and Maintenance of Applications in 3rd and 4th Generation Languages," *Information & Management*, 18 No3: 131 - 141 (March 1990).

33. Low, Graham C. and D. Ross Jeffery. "Function Points in the Estimation and Evaluation of the Software Process," *IEEE Transactions on Software Engineering*, 16 No1: 64 - 71 (January 1990).

34. Mukhopadhyay, Tridas, Michael J. Prietula, and Steven S. Vicinaza. "Examining the Feasibility of a Cased-Based Reasoning Model for Software Effort Estimation," *MIS Quarterly*, 16 No2 : 155 - 171 (June 1992).

35. NeSmith, Robert E II. *A Study of Maintenance Costs of Air Force Large Scale Computer Systems*. MS thesis, AFIT/GSM/LSM/86S-15. School of Systems and Logistics, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, September 1986 (AD-A174454).

36. Neter, John and others. *Applied Linear Regression Models*. Boston MA: Irwin, 1989.

37. Oman, Paul and Jack Hagemeister. "Metrics for Assessing a Software System's Maintainability," *Proceedings of Conference on Software Maintenance*. 337 - 344. Washington DC: IEEE Computer Society Press, 1992.

38. Ourada, Gerald L. *Software Cost Estimating Models: A Calibration, Validation, and Comparison*. MS Thesis, AFIT/GSS/LSY/91D-11. School of Systems and Logistics, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, December 1991 (AD-A246677).

39. Page-Jones, Melir. *Structured Systems Design*, Englewood Cliffs NJ: Prentice-Hall, Inc., 1988.

40. Penedo, Maria H. and Christine Shu. "Acquiring Experience With the Modelling and Implementation of the Project Life-Cycle Process: The PMDB Work," *Software Engineering Journal*, 6 No5: 259 - 274 (September 1991).

41. Price, Gordon, Bryce Ragland, and Gregory Daich. *Source Code Static Analysis Technologies Report, Vol I*, Hill AFB UT: Software Technology Support Center (STSC), March 1993.

42. Price, Gordon, Bryce Ragland, and Gregory Daich. *Source Code Static Analysis Technologies Report, Vol II*, Hill AFB UT: Software Technology Support Center (STSC), March 1993.

43. Rehg, Virgil. "What Should Cost Estimators Know about TQM?" *National Estimator*, Spring: 32 - 39 (1991).

44. Reifer, Donald J. "Asset-R: A Function Point Sizing Tool for Scientific and Real-Time Systems," *Journal of Systems and Software*, 11 No3: 159 - 171 (March 1990).

45. Revised Intermediate COCOMO (REVIC). Version 9.0, IBM, 244k, disk. Computer Software and Manual. Raymond L. Kile, 6 April 1992.

46. Robson, D. J., and others. "Approaches to Program Comprehension," *Journal of Systems and Software*, 14: 79 - 84 (February 1991).

47. Rubey, Raymond J. *Software Management Guide*, Hill AFB UT: Software Technology Support Center (STSC), April 1992.

48. Rumbaugh, James and others. *Object-Oriented Modeling and Design*. Englewood Cliffs NJ: Prentice Hall, 1991.

49. Schwartz, Evan I. "Turning Software from a Black Art into a Science," *Business Week*, Special Issue: 80 - 81 (July 1992).

50. Silver, Aaron N. and Joseph D. Suhr. *Technical Report Software Cost Estimation Study, CER Model Baseline Report*. Contract N00014-85-C-0892. Denver CO: Martin Marietta Denver Aerospace Corporation. September 1987 (AD-B116 049).

51. Silver, Aaron N. and others. *SASET User's Guide (Multiple CPCI Enhancement) Software Architecture, Sizing, and Estimating Tool.* Contract N00014-85-C-0892. Denver CO: Martin Marietta Denver Aerospace Corporation, February 1990.

52. Sittenauer, Chris and Mike Olsen. "Time to Re-engineer?" *Crosstalk*, No32: 7 - 10 (March 1992).

53. Stewart, Rodney D. and Richard M. Wyskida. *Cost Estimator's Reference Manual*, John Wiley & Sons, 1987.

54. Symons, Charles R. *Software Sizing and Estimating Mk II FPA (Function Point Analysis).* West Sussex England: John Wiley & Sons Ltd., 1991.

55. SYSCON Corporation. *Avionics Software Support Cost Model: Final Report, Vol I, September 1980 - November 1982.* Contract F33515-80-C-1157. Washington DC: SYSCON Corporation, 1 February 1983 (AD-A128523).

56. System Evaluation and Estimation of Resources (SEER). Version 3.0, IBM, 603k, disk. Computer Software and Manual. Marina del Rey CA: Galorath Associates, Inc., 15 March 1991.

57. Talbot, John and others. *Post Deployment Software Support Process.* Report from Sacramento Air Logistics Center Process Action Team. McClellan AFB CA: Sacramento ALC, 31 October 1990.

58. Thibodeau, Robert. *An Evaluation of Software Cost Estimating Models*. Contract F30602-79-C-02244. Huntsville AL: General Research Corporation, 10 April 1981 (AD-A104226).

59. Törn, Aimo A. "Models of Software Accumulation," *Journal of Systems and Software*, 12 No1: 39 - 42 (April 1990).

60. van Genuchten, Michiel J. I. M. and Hans J. A. H. M. Koolen. "Applications on the Use of Software Cost Models," *Information & Management*, 21 No1: 37 - 44 (August 1991).

61. Waina, R. B. and others. *Predictive Software Cost Model Study: Final Technical Report, Vol I, 2 April 1979 - 2 June 1980*. Contract F33615-79-C-1734. Canoga Park CA: Hughes Aircraft Company, June 1980 (AD-A088476).

62. Waina, R. B. and others. *Predictive Software Cost Model Study: Final Technical Report, Vol II, 2 April 1979 - 2 June 1980*. Contract F33615-79-C-1734. Canoga Park CA: Hughes Aircraft Company, June 1980 (AD-A088477).

63. Yuen, Chong Hok. "A Statistical Rationale for Evolution Dynamics Concepts," *IEEE Conference on Software Maintenance - 1987*. 156-164. Washington DC: Computer Society Press of the IEEE, 1987.

Vita

Captain Ronald L. Warner, Jr., was born on 2 May 1960 in Okinawa Japan. He graduated from Whitesboro Senior High School in 1978 and attended the U.S. Air Force Academy, graduating in 1982 with a Bachelor of Science in Electrical Engineering with Academic Honors and Distinction. Upon graduation, he received a regular commission in the USAF and attended Undergraduate Pilot Training (UPT) at Williams AFB, Arizona. His subsequent flying assignment, Castle AFB, California, included duties as a copilot and aircraft commander for the 924th Air Refueling Squadron. His next assignment, at Wright-Patterson AFB, Ohio, encompassed flying as a dual-qualified instructor research pilot in the NKC-135 and the NT-39 for the 4950th Test Wing. During this tour, he was selected as the deputy test program manger for Tanker conversion, a KC-135 aircraft modification project that provided a calibrated water spray for airborne icing tests. He also supported several test missions world-wide including the shuttle launch of the Magellan space vehicle and obtained a Master of Science Degree in Aerospace Engineering from the University of Dayton. Captain Warner departed the Test Wing in April 1992 to attend the Air Force Institute of Technology's graduate program in Systems Software Management.

Permanent Address: 8901 Gardengate Dr.
Huber Heights OH 45424

Vita

Captain Darrell L. Wright was born on 1 March 1961 in Cheyenne Wyoming. He graduated form East High School in Cheyenne in 1979. He received an Air Force ROTC scholarship and attended Nebraska Wesleyan University, Lincoln, Nebraska, graduating with a Bachelor of Science in Physics (With Emphasis in Computer Technology) in June 1983. Upon graduation, he was commissioned as a reserve officer and began active duty service at the Air Force Geophysics Laboratory, Hanscom AFB, Massachusetts. He was a program control officer and deputy program manager for the Missile Surveillance Technology (MST) program and later, the Cryogenic Infrared Radiance Instrumentation for Shuttle (CIRRIS-1A) program. He also served as the Lab's cost estimation focal point. He transferred to Wright-Patterson AFB in October 1987 and served as Technology Integration Manager for the Integrated Electronic Warfare System (INEWS) program and later, the SEEK SPARTAN program. He was reassigned to the F-16 System Program Office at Wright-Patterson in 1991 where he was the Avionics Integration Manager for several F-16 avionics systems and for the procurement of F-16 unique pilot training materials and equipment. He entered the School of Systems and Logistics, Air Force Institute of Technology, in May 1992.

Permanent Address: 2164 Knoll Dr
Beavercreek OH 45431

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	December 1993	Master's Thesis

4. TITLE AND SUBTITLE
DEMONSTRATION OF IMPROVED SOFTWARE SUPPORT LABOR ESTIMATION FOR AIR FORCE OPERATIONAL FLIGHT PROGRAMS THROUGH FUNCTIONAL ORIENTATION

5. FUNDING NUMBERS

6. AUTHOR(S)
Ronald L. Warner Jr., Captain, USAF
Darrell L.Wright, Captain USAF

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Air Force Institute of Technology
Wright-Patterson AFB OH 45433-6583

8. PERFORMING ORGANIZATION REPORT NUMBER
AFIT/GSS/LAS/93D-7

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
None.

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION / AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

This study demonstrated two approaches to improve current software support effort estimation models for aircraft software. Both approaches involved a functional orientation not used by existing models. The first approach demonstrated how to orient a model to reflect the block change cycle modification process and how to represent support effort changes over time in order to improve effort estimation accuracy. Current software models do not reflect the support environment or the temporal characteristics of aircraft software support. The second approach demonstrated how to calibrate a model by properly selecting source data in order to increase accuracy. Support calibration is not addressed by current models. A literature search affirmed the validity of both approaches and the methodology. In addition, a standard description of the block change cycle was developed and validated. A prototype estimation model was derived from the COCOMO model and included a unique support calibration. Data was obtained from Air Force Software Support Centers but was unusable, so data was generated from the prototype for the demonstration. A method that was developed to compare the prototype with current models demonstrated that the prototype is an acceptable model.

14. SUBJECT TERMS	15. NUMBER OF PAGES
Software Life Cycle Cost, Software Cost Estimation, Software Support, COCOMO, REVIC, SASET, Cost Models, Software Maintenance	251
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL

AFIT Control Number AFIT/GSS/LAS/93D-7

AFIT RESEARCH ASSESSMENT

The purpose of this questionnaire is to determine the potential for current and future applications of AFIT thesis research. Please return completed questionnaires to: DEPARTMENT OF THE AIR FORCE, AIR FORCE INSTITUTE OF TECHNOLOGY/LAC, 2950 P STREET, WRIGHT PATTERSON AFB OH 45433-7765

1. Did this research contribute to a current research project?

a. Yes b. No

2. Do you believe this research topic is significant enough that it would have been researched (or contracted) by your organization or another agency if AFIT had not researched it?

a. Yes b. No

3. The benefits of AFIT research can often be expressed by the equivalent value that your agency received by virtue of AFIT performing the research. Please estimate what this research would have cost in terms of manpower and/or dollars if it had been accomplished under contract or if it had been done in-house.

Man Years _____ $ _____

4. Often it is not possible to attach equivalent dollar values to research, although the results of the research may, in fact, be important. Whether or not you were able to establish an equivalent value for this research (3, above) what is your estimate of its significance?

a. Highly Significant b. Significant c. Slightly Significant d. Of No Significance

5. Comments

_____ _____

Name and Grade Organization

_____ _____

Position or Title Address

DEPARTMENT OF THE AIR FORCE
AFIT/LAC Bldg 641
2950 P St
45433-7765

OFFICIAL BUSINESS

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS MAIL PERMIT NO. 1006 DAYTON OH

POSTAGE WILL BE PAID BY U.S. ADDRESSEE

Wright-Patterson Air Force Base

AFIT/LAC Bldg 641
2950 P St
Wright-Patterson AFB OH 45433-9905